INFORMATION TECHNOLOGY

113 A Direct Search Method Applied to a Molecular Structure Problem
   W Pullan

121 Extending PROTEAN with the Partitioning Method
   R Lai

128 Genetic Algorithm for Fine-Tuning Fuzzy Rules for the Cart-Pole Balancing System
   G D Finn

138 The Power of List Ranking on a Reconfigurable Array of Processors with Wider Bus Networks
   T W Kao, S J Horng

SPECIAL FEATURES

120 Call for Papers — Special Issue on Multimedia Storage and Archiving Systems

Published for Australian Computer Society Incorporated
Print Post Approved PP255003/01660
Please record my new address as shown above. I enclose a cheque or bank draft for $50.00 local, $65.00 overseas.

Please record my new address as shown above. I attach the mailing label for the last received issue.

*ATTACH LABEL HERE*
A Direct Search Method Applied to a Molecular Structure Problem

W Pullan
Central Queensland University, Rockhampton, Queensland 4702
W.Pullan@cqu.edu.au

In a previous paper, Benke and Skinner (1991) described a direct search method for locating the global optimum of a multimodal function. This search method used an adaptive probabilistic algorithm and was suitable for any function of many variables subject to arbitrary constraints. This paper describes an extension to this search method which generally improves the rate of convergence to the global optimum. The effect of this extension is evaluated using a number of standard multimodal test functions and a molecular structure problem.

Keywords and Phrases: Global optimisation, direct search procedures, adaptive random search, numerical methods.

CR Categories: F.2.1 (Numerical Algorithms and Problems), G.1.6 (Optimisation), I.2.8 (Problem Solving, Control Methods and Search).

INTRODUCTION
If you want to plan an efficient bus route, optimise the positions of integrated circuits on a silicon chip, or predict the lowest energy configuration of a cluster of atoms, the problem reduces to finding the minima of specified functions. The significance of the optimisation field is reflected in the number of papers written on the subject. Over 300 articles on one method, simulated annealing, have been published since 1988.

Mathematically, the global optimisation problem can be stated as:

$$\min_{x \in S} f(x)$$

where $f(x): \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous function and $S \subseteq \mathbb{R}^n$. There appears to be no one "perfect" algorithm that will solve every optimisation problem. Instead, a host of tailored methods have evolved, each suited to specific tasks. For example, golden section search can be used if the function to be minimised (or maximised) has only one dependent variable and its derivative is unknown. If derivatives can be calculated, conjugate gradient methods or variable metric methods may be used. For such methods, however, the final value found is related to the starting point, so these techniques are often trapped at a local extremum. Techniques which broadly come under the heading of "Random Search" can be used to minimise the chance of entrapment.

Random search techniques currently available are many and varied. Among the more important are simulated annealing, an extension of a Monte Carlo method developed to determine the equilibrium states of a collection of atoms at a given temperature $T$, and genetic algorithms, which are based on the mechanics of natural selection and genetics. Benke and Skinner (1991) described a random search technique for locating the global optimum of any multimodal, multivariate function subject to arbitrary constraints. Their technique, an adaptive probabilistic search, referred to in this paper as APS, is based on a simple model for noise reduction and uses an iterative method for averaging random perturbations in the parameter estimates. No prior assumptions about the search domain, nor about the continuity, differentiability or modality of the function are required by the APS algorithm.

This paper extends APS into a search method, referred to as APSE (for APS Extended), and evaluates the effect of this extension. Section 2 describes the APS algorithm, the extension applied to APS to produce APSE and a proof of convergence for APSE. Section 3 evaluates both APS and APSE using standard test functions and also investigates the relationship between problem dimensionality and the number of function evaluations required by APSE to find an extremum. Section 4 presents an overview of the particular problem addressed in this paper, that of determining the minimum energy configurations of clusters of atoms (microclusters) and also presents the results obtained from applying APSE to this problem. Section 5 contains discussion on the results obtained and possible extensions to this study.
2 APS AND APSE ALGORITHMS
The basic algorithm for APS (Benke and Skinner, 1991), to minimise a function \( f(x) \) (assumed to be negative), is as follows:

**Initial Step:**
Set \( k = 0 \).
Generate random vector \( x^* \)
within specified constraints.
Set \( f^* = f(x^*) \).

**Iterative Step:**
(i) Generate random vector \( x^*_1 \)
within specified constraints.
Set \( f^*_1 = f(x^*_1) \).
(ii) Set \( x^*_2 = \frac{(f^*_1 x^*_1 + f^*_2 x^*_2)}{(f^*_1 + f^*_2)} \).
Set \( f^*_2 = f(x^*_2) \).
(iii) Set \( x^*_1 = \text{arg min} (f^*_1, f^*_2, f^*_2) \).
Set \( f^*_1 = f(x^*_1) \).

**Stopping Criteria:**
If termination criterion reached
Optimum vector is \( x^*_1 \)
Optimum function value is \( f^*_1 \)
stop.
else
Set \( x^*_1 = x^*_1 \)
Set \( f^*_1 = f(x^*_1) \)
Set \( k = k + 1 \)
Repeat Iterative Step.

As suggested in Benke and Skinner (1), the rate at which APS converges to a minimum function value can be accelerated by systematically reducing the search domain during the course of the optimisation. That is, if after some number of iterations \( x^*_1 \) remains unchanged, then the domain over which \( x^*_1 \) is generated is progressively reduced in size. After the first reduction step, the centre of the new domain is taken as the current \( x^*_1 \) and, whenever \( x^*_1 \) is changed, the current search domain is recentred at \( x^*_1 \). Obviously the danger inherent in reducing the search domain is that the chance of finding a local rather than a global minimum is increased. To reduce this possibility, the extension was implemented in APSE as follows:

— if APSE remains at \( x^*_1 \) for a number of function evaluations then the search domain is reduced;
— this reduction occurs repeatedly for a specified number of iterations and then the process reverses, and the search domain is expanded in steps until the original search domain is reached; and
— this cycle of reduction/expansion is repeated a number of times and has the effect of biasing the search around \( x^*_1 \) but does not ignore more distant regions of the search domain.

The parameters that control the search method are:

— \( \lambda \), the number of \( x^*_1 \) repeats required before a reduction/expansion of the search domain is performed;
— the amount by which the search domain is reduced (currently in APSE, each dimension of the search domain is reduced by a factor of 2);
— \( \beta \), the number of reductions performed before the expansion phase is entered;
— the amount by which the search domain is expanded (currently in APSE, each dimension of the search domain is increased by a factor of 2);
— the number of reduction/expansion cycles to be performed before the search terminates. For the test functions used in this paper, as the global minimum values were all known in advance, searches were terminated when the desired accuracy had been achieved.

Clearly the best values to be used for \( \lambda \) and \( \beta \) depend on the nature of the function being optimised.

Given \( \lambda \) and \( \beta \) as defined above plus the following definitions,

— \( c_1 \), the number of current \( x^*_1 \) repeats;
— \( c_2 \), the current count of the number of reduction/expansion operations performed on the search domain \( S \);
— \textit{Reducing}, a boolean flag which is True when the search is in the phase of reducing the search domain;
— \textit{reduce}(S), a function which reduces the search domain by one half in all dimensions;
— \textit{expand}(S), a function which doubles the search domain in all dimensions; and
— \textit{recentre}(S, x), a function which recentres the search domain around \( x \).

then the APSE algorithm, to minimise a function \( f(x) \) (assumed to be negative), is as follows:

**Initial Step:**
Set \( k = 0 \).
Generate random vector \( x^*_1 \)
within specified constraints.
Set \( f^*_1 = f(x^*_1) \).
Set \( c_1 = 0 \).
Set \( c_2 = 0 \).

**Iterative Step:**
(i) Generate random vector \( x^*_2 \)
within specified constraints.
Set \( f^*_2 = f(x^*_2) \).
(ii) Set \( x^*_3 = \frac{(f^*_1 x^*_1 + f^*_2 x^*_2)}{(f^*_1 + f^*_2)} \).
Set \( f^*_3 = f(x^*_3) \).
A DIRECT SEARCH METHOD APPLIED TO A MOLECULAR STRUCTURE PROBLEM

(iii) If \( f_i = \text{Min}(f_{i1}', f_{i2}', f_{i3}') \) and \( c_i = \lambda \)
    \[ c_i = 0. \]

If Reducing
    If \( c_i < \beta \)
        \[ S = \text{reduce}(S) \]
        \[ c_i = c_i + 1 \]
    else
        Reducing = False
        \[ S = \text{expand}(S) \]
        \[ c_i = c_i - 1 \]
else
    If \( c_i > 0 \)
        \[ S = \text{expand}(S) \]
        \[ c_i = c_i - 1 \]
    else
        Reducing = True
        \[ S = \text{reduce}(S) \]
        \[ c_i = c_i + 1 \]
\[ \text{else} \]

(iv) \[ x_i^* = \text{argmin}(f_i, f_{i1}', f_{i2}', f_{i3}'). \]

Set \( f_i = f(x_i^*) \)
recentre\((S, x_i^*)\)

Stopping Criteria:
If termination criterion reached
    Optimum vector is \( x_i^* \)
    Optimum function value is \( f_i^* \)
    stop.
else
    \[ x_i^{*+1} = x_i^* \]
    \[ f_i^{*+1} = f_i^* \]
    \[ k = k + 1 \]
Repeat Iterative Step.

Intuitively, the reduction/expansion of the current search domain assists convergence of the search because:

— when \( f_i \) is considerably greater than the global minimum, there is a lower probability that \( c_i \) will reach the value required for a reduction in the search domain. Therefore the rate at which the reduction/expansion cycle takes place will be relatively slow. That is, the search will, on average, be more localised around the current \( x_i \).

2.1 Convergence of APSE
The APSE algorithm can be reduced to the following conceptual algorithm:

Step 1: Set \( k = 0 \). Find \( x_i^0 \) in \( S \).
(APSE, Initialise Step)

Step 2: Generate \( x_i^1 \) from the sample space \((R^n, \mu_i)\).
(APSE, Iteration Step i)

Step 3: Generate \( x_i^2 \) as \( G(x_i^1, x_i^2) \).
(APSE, Iteration Step ii)

Step 4: Set \( x_i^{+1} = D(x_i^1, x_i^2, x_i^3) \).
(APSE, Iteration Steps iii & iv)

Step 5: Choose \( \mu_{x_i} \), set \( k = k + 1 \) and return to Step 2.
(APSE, Stopping Criteria)

where:
— the map \( G \) has domain a subset of \( S \times S \) and range \( S \);
— the map \( D \) has domain a subset of \( S \times S \times S \) and range \( S \); and
— the \( \mu_i \) are probability measures corresponding to distribution functions defined on \( R^n \).

Given the following definition and conditions:

Definition I.
If the global minimum value of \( f \) is denoted by \( x^* \), then the \( \varepsilon \) –optimal region for \( f \) with \( \varepsilon > 0 \), is defined by
\[ R_{\varepsilon} = \{ x \in S | f(x) < x^* + \varepsilon \} \]

Condition I.
The map \( D \) satisfies:
\[ f(D(x_1, x_2, x_3)) \leq f(x_i) \]
and if \( x_i \in S \),
\[ f(D(x_1, x_2, x_3)) \leq f(x_i) \]

Condition II.
For any subset \( A \) of \( S \) with \( v(A) > 0 \), we have that
\[ \prod_{x_i \in A} [1 - \mu_{x_i} (A)] = 0 \]
where:
\( v(A) \) is the \( n \)-dimensional volume of the set \( A \).
\( \mu_{x_i} (A) \) is the probability that the randomly chosen \( x_i \) or \( x_i^* \) is in \( v(A) \).

This condition means that, given any subset \( A \) of \( S \) with positive “volume”, the probability of repeatedly missing the set \( A \), when generating the random samples \( x_i \) and \( x_i^* \), must be zero. Clearly this condition is satisfied by APS and APSE (because the cyclic nature of the reduction/expansion cycle ensures that no portion of the search region is ever discarded).

As APSE satisfies conditions I and II, the proof of convergence for APSE is identical to that in Solis and Wets (1981) for global search. We present this now:
APSE Convergence Theorem
Assume $f$ is a measurable function, $S$ a measurable subset of $R^n$ and Conditions I and II above are satisfied. Let $R_e$ denote the $\varepsilon$-optimal region for $f$ and $\{x^k\}_{k=0}^\infty$ be a sequence generated by APSE. Then

$$\lim_{k\to\infty} P[x^k \in R_e] = 1$$

where $P[x^k \in R_e]$ is the probability that at step $k$, the point $x^k$ generated by APSE is in $R_e$.

Proof: From Condition I, it follows that $x^k$ or $x_{k+1}$ or $x^k_{\infty}$ implies that $x^k \in R_e$ for all $k \geq k + 1$. Thus

$$P[x^k \in S \mid R_e] = \prod_{i=0}^{k-1} \left( 1 - \mu_i(R_e) \right)$$

$$= (1 - \mu_k(R_e)) \prod_{i=0}^{k-1} (1 - \mu_i(R_e))$$

$$\leq \prod_{i=0}^{k-1} (1 - \mu_i(R_e))$$

Therefore

$$P[x^k \in R_e] = 1 - P[x^k \in S \mid R_e]$$

$$\geq 1 - \prod_{i=0}^{k-1} (1 - \mu_i(R_e))$$

and hence

$$1 \geq \lim_{k\to\infty} P[x^k \in R_e] \geq 1 - \lim_{k\to\infty} \prod_{i=0}^{k-1} (1 - \mu_i(R_e)) = 1$$

where the last equality follows from Condition II.

3 TEST FUNCTIONS
Tests to determine the effectiveness of the extension to APS were made using a number of test functions. For each test function, the basic test procedure was:

— perform a series of APSE runs, using different values of $\lambda$ and $\beta$, to determine the best values for a particular test function; and

— run 100 trials for each test function using APS and then repeat these 100 trials using APSE with the optimum values for $\lambda$ and $\beta$ determined above. Each trial used a random starting point and terminated when a function value within 0.0001 of the global minimum was obtained.

As the molecular structure problem has high dimensionality ($3 \times N - 6$, where $N$ is the number of atoms), a series of tests were also performed to evaluate the effect of problem dimensionality on APSE. The results obtained are discussed in Section 3.5.

3.1 Beale Test Function
Minimise $\sum_{i=1}^{3} U_i^2$, where $U_i = c_i - x_i (1 - x_i^2)$, with $c_1 = 1.5$, $c_2 = 2.25$, $c_3 = 2.625$, $-10 \leq x_i \leq 10$, and using a random starting point (Pierre, 1986). This function has a narrow curving valley which approaches the line $x_3 = 1$ and a minimum at $(3, 0, 5)$. While this function has only one minimum, it does test the ability of APS and APSE to “track” along a narrow valley. Figure 1 shows the result of varying $\lambda$ and $\beta$ on the number of function evaluations required by APSE to find the minimum. By inspection it can be seen that $\lambda = 6$ and $\beta = 3$ result in the lowest number of function evaluations for APSE. Figure 2 shows the result of 100 trials for the non-reducing version of APS and APSE (using $\lambda = 6$ and $\beta = 3$).

3.2 Goldstein-Price Test Function
Minimise

$$[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times$$

$$[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

where $-2 \leq x_i \leq 2, i = 1, 2$, using a random starting point (Törn and Zilinskas, 1987). This function has a global minimum of 3 when $x = (0, -1)$ and 4 local minima within the region of interest.
3.3 Rastrigin Test Function
Minimise \[ x_1^2 + x_2^2 - \cos 18x_1 - \cos 18x_2 \]
where \(-1 \leq x \leq 1\), using a random starting point (Törn and Zilinskas, 1987). This function has a global minimum of \(-2\) when \(x = (0, 0)\) and some 50 local minima arranged in a lattice configuration within the region of interest.

3.4 Griewank-2 Test Function
Minimise
\[ \sum_{i=1}^{d} \frac{x_i^2}{4} - \prod_{i=1}^{d} \cos(x_i / \sqrt{i} \sin(\lambda x_i^2 + 1) \]
where \(n = 2\), \(d = 200\) and \(-100 < x < 100\), using a random starting point (Törn and Zilinskas, 1987). This function has a global minimum of \(0\) when \(x = 0\) and some 500 local minima in the region of interest.
A DIRECT SEARCH METHOD APPLIED TO A MOLECULAR STRUCTURE PROBLEM

1E+20
1E+12
10000
Figure 9: Number of Minima for $F(x)$.

3.5 Variable Dimensional Test Function
The variable dimensional test function used was:

$$F(x) = -5 \prod_{k=1}^{n} \sin(x_k) - \prod_{k=1}^{n} \sin(5x_k)$$

which has a global minimum value of $-6.0$ at $x_k = \pi/2, k = 1, ..., n$. The test problem is to minimise $F(x)$ where $0 \leq x_k \leq \pi, k = 1, ..., n$. The number of local minima for this test function is plotted in Figure 9 and is given by Zabinsky Graesser, Tuttle and Kim (1992):

$$\sum_{i=0}^{n} \frac{n!}{(n-2i)(2i)!} 3^{n-2i/2i}$$

The average number of function evaluations taken by APS and APSE, as a function of dimension, is shown in Figure 10. Each sample point is the average of five separate runs with the starting point for each run randomly chosen. Runs are terminated when the objective function is within 0.001 of the known global minimum.

As can be seen from Figure 10, at least up to 25 dimensions and as a function of dimension, APSE requires considerably fewer function evaluations than APS. This is a desirable feature in any search technique used to determine the minimum energy configurations of the microclusters of the following section, as the dimensionality of the problem is high, even for small microclusters.

4 MICROCLUSTERS
Global optimisation problems exist in a number of sciences. The protein folding problem in Computational Biology and molecular structure problems in Theoretical Chemistry are just two examples of problems which are currently receiving attention by researchers in the global optimisation field. This section presents an overview of a molecular structure problem, that of determining the minimum energy configuration of a group of neutral atoms (a microcluster) and also describes the results obtained by applying APSE to this problem.

Microclusters are aggregates of atoms or molecules, sufficiently small so that a significant proportion of these units are present on the surface of the microcluster. The optimisation problem associated with microclusters is:

**Given** $N$ particles, interacting with two-body central forces, find the configuration in three-dimensional Euclidean space for which the total potential energy attains its global minimum.

Simplifications normally used are:

— Many-body and angle dependent interactions are ignored;
— Quantum effects are not taken into account; and
— All particles are assumed to be spherical and the same size.

Using these simplifications, the potential energy of an $N$ atom microcluster can be written as:

$$V = \sum_{i=1}^{N} \sum_{j=i+1}^{N} v(r_{ij})$$

where $r_{ij}$ is the Euclidean distance between atoms $i$ and $j$ and $v(r_{ij})$ is the pair potential between atoms $i$ and $j$. Even with these simplifying assumptions, the minimisation of the potential energy of a microcluster is very difficult to achieve as it as a nonconvex optimisation problem involving numerous local minima.

Hoare (1979) showed that the number of local minima of an $N$-atom microcluster appears to have the form $g(N) \equiv \exp(-2.5176 + 0.3572N + 0.0286N^2)$. This function is plotted in Figure 11 along with the counts of local minima determined by an exhaustive search of microclusters where $N \leq 13$. Comparing Figure 11 with Figure 9 shows that, for a comparable number of dimensions (each atom has three degrees of freedom), the variable dimensional test function has more local minima than the microcluster problem.
A DIRECT SEARCH METHOD APPLIED TO A MOLECULAR STRUCTURE PROBLEM

Figure 11: Distinct local minima for Lennard-Jones microclusters.

A number of optimisation methods have been applied to the microcluster problem. They include lattice optimisation/relaxation techniques (Northby 1987; Xue 1984), simulated annealing (Wille 1987; Xue 1994; Navon, Brown and Roberson, 1990), DCTransformation/GOP Algorithm (Marans and Floudas, 1992), Genetic Algorithm (Judson, Colvin, Meza, Huffer and Gutierrez, 1992; Judson, Colvin, Meza, Huffer and Gutierrez, 1991) and the Diffusion Equation Method (Kostrowicki, Piela, Cherayil and Scheraga, 1991). Of these, the lattice optimisation/relaxation techniques have been most successful and currently provide most of the definitive results. However, as considerable experimental evidence was used in determining the starting configurations for each lattice optimisation, the lattice optimisation/relaxation technique cannot be considered a general global optimisation method.

APSE was applied to microclusters using the reduced form of the Lennard-Jones potential, \( v(r) = r^{-12} - 2r^{-6} \), for microclusters of \( N \) atoms with \( N \) in the range 3 \( \leq N \leq 20 \). This reduced form of the Lennard-Jones potential is commonly used in studies of microclusters and has the form shown in Figure 12.

Figure 12: Reduced Lennard-Jones potential.

Because of the difficulty of solving the microcluster problem, APSE was used in an iterative manner. Each run of APSE started from a two atom cluster (which has the obvious solution of two atoms separated by a distance of 1 unit) and constructed larger microclusters by adding a single atom, to the minimum configuration found for the \( N-1 \) atom microcluster, to produce an \( N \) atom microcluster. To determine the optimal configurations, APSE optimisations were performed as each new atom was added.

As stated above, the global minimum structure for \( N = 2 \) is two atoms, distance 1 unit apart. For \( N = 3 \), the global minimum structure is an equilateral triangle where each side is of length 1 unit. For \( N = 4 \), the 4 atoms form a tetrahedron where each edge is of length 1 unit. For \( N = 5 \), a triangular bipyramid, slightly contracted along the symmetry axis and distended in the symmetry plane yields the global minimum configuration. For \( N = 6 \), a regular octahedron with slightly contracted sides yields the global minimum configuration and, for \( N = 7 \), the regular icosahedron (pentagonal bipyramid) with slightly distended edges and contracted axial distances is the structure producing the global minimum for potential energy. For \( N > 7 \), minimum energy structures are based on icosahedral lattice structures.

Table 1 presents a comparison of the results, obtained for single runs of APSE, with those obtained by Northby (1987), Xue (1994) and Maranas and Floudas (1992). As can be seen, at least to two decimal places, there is agreement between the results obtained by APSE and other methods. However the number of function evaluations, when compared with those needed for the test function in Section 3.5, clearly demonstrate the complexity of the potential energy surface for microclusters.

Table 1: Minimum energies for Lennard-Jones microclusters.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-3.00</td>
<td>-3.00</td>
<td>909</td>
</tr>
<tr>
<td>4</td>
<td>-6.00</td>
<td>-6.00</td>
<td>4361</td>
</tr>
<tr>
<td>5</td>
<td>-9.10</td>
<td>-9.10</td>
<td>5665</td>
</tr>
<tr>
<td>6</td>
<td>-12.71</td>
<td>-12.71</td>
<td>103334</td>
</tr>
<tr>
<td>7</td>
<td>-16.51</td>
<td>-16.51</td>
<td>12166</td>
</tr>
<tr>
<td>8</td>
<td>-19.82</td>
<td>-19.82</td>
<td>20046</td>
</tr>
<tr>
<td>9</td>
<td>-24.11</td>
<td>-24.11</td>
<td>11462</td>
</tr>
<tr>
<td>11</td>
<td>-32.77</td>
<td>-32.77</td>
<td>6891</td>
</tr>
<tr>
<td>12</td>
<td>-37.97</td>
<td>-37.97</td>
<td>76515</td>
</tr>
<tr>
<td>13</td>
<td>-44.33</td>
<td>-44.33</td>
<td>122150</td>
</tr>
<tr>
<td>14</td>
<td>-47.85</td>
<td>-47.85</td>
<td>55860</td>
</tr>
<tr>
<td>15</td>
<td>-52.32</td>
<td>-52.32</td>
<td>9943</td>
</tr>
<tr>
<td>16</td>
<td>-56.82</td>
<td>-56.82</td>
<td>252949</td>
</tr>
<tr>
<td>17</td>
<td>-61.32</td>
<td>-61.32</td>
<td>106052</td>
</tr>
<tr>
<td>18</td>
<td>-66.53</td>
<td>-66.53</td>
<td>71467</td>
</tr>
<tr>
<td>19</td>
<td>-72.66</td>
<td>-72.66</td>
<td>236523</td>
</tr>
<tr>
<td>20</td>
<td>-77.18</td>
<td>-77.18</td>
<td>114344</td>
</tr>
</tbody>
</table>

5 DISCUSSION

Molecular structure problems, particularly for larger microclusters, have a large number of dimensions and a complicated potential energy surface. The APSE search method described here has been successful with small numbers of atoms but processing requirements for larger numbers are likely to be substantial.
More success would appear to be possible using APSE if the nature of the potential energy surface was better understood and the APSE parameters adjusted accordingly. In addition, parallel versions of APSE should make possible the determination of minimum energy configurations for larger microclusters than those reported here.

ACKNOWLEDGEMENTS

The author thanks Graham Wood and the anonymous referees for their critical and helpful comments in preparing this paper.

REFERENCES


BIOGRAPHICAL NOTES

Wayne Pullan is a senior lecturer in the Department of Mathematics and Computing at Central Queensland University. His research interests include software engineering and global optimisation.

CALL FOR PAPERS

**AUSTRALIAN COMPUTER JOURNAL**

**Special Issue on Multimedia Storage and Archiving Systems**

Advances in technology have resulted in huge archives of sounds, images and videos in diverse application areas such as medicine, remote sensing, industry, engineering, entertainment, education and on-line information services. To exploit the value of the collection, there is a need to effectively organise multimedia data for storage and retrieval. This special issue of Australian Computer Journal is devoted to addressing various aspects of the development and applications of multimedia archiving systems.

Papers are solicited in all areas of multimedia data storage and archival systems including, but not limited to:

- image representation
- image and video characterisation for indexing
- feature-based retrieval methods for multimedia data
- audio, image or video databases
- neural network, fuzzy and information theoretic methods for image and video indexing
- storage design and management for image and video archiving
- object-oriented design and user interfaces
- automatic multimedia authoring
- applications and systems: digital libraries, video on demand, tele-medicine, remote sensing, GIS, education, entertainment, interactive TV, CAD etc.

Schedule:
- Deadline for submission: 10 March 1997
- Notification of decision: 30 June 1997
- Publication: August 1997

Submission:
Send four copies (general guidelines can be found in any issue of Australian Computer Journal and full instructions to authors can be obtained from Dr Jesse Jin) to either of the following Guest Editors:

- Dr Jesse Jin
  School of Computer Science and Engineering
  University of New South Wales
  Sydney NSW 2052, Australia

- Prof Bala Srinivasan
  Department of Computer Technology
  Monash University
  900 Dandenong Road
  Victoria 3145, Australia
Extending PROTEAN with the Partitioning Method

R Lai
School of Computer Science and Computer Engineering,
La Trobe University, Victoria, Australia 3083

Reachability Analysis (RA) is the most popular protocol verification technique as it can verify a number of protocol properties, like deadlock freeness, livelock freeness and boundedness; and it can be easily automated. However, it is also very well-known that the state space explosion problem of RA limits its use in complex situations. The usefulness of existing verification tools in verifying complex protocols is very much restricted by this problem. Various approaches have been proposed to tackle the problem; and, so far, there have been very rare reports on how these approaches can be implemented. Without actually incorporating these techniques in a verification tool, the proposed methods remain theoretical.

To date, there are a number of communication verification tools available but they have hardly been used successfully in complex cases, as they are handicapped by the state space explosion problem. The relief strategy we are particularly interested in is the partitioning method (Tridgell, 1987). It partitions a Petri Net (Peterson, 1981) by cutting a line that goes through some transitions while preserving the characteristics of the main net and then each of the subnets will be analysed separately. To build a tool from scratch to include this algorithm will require many man years of software development effort. To speed up the process, this method is to be incorporated in an existing tool, PROTEAN (Billington, 1988).

PROTEAN was written in Pascal. It consists of a number of modules; one of them is the reachability analysis module which generates a Reachability Graph (RG) automatically. This module is very complex and comprises about 1,600 lines of codes. To incorporate the partitioning technique in this module is a difficult task as it requires substantial changes to the code and the algorithms in order to generate new RGS.

This paper describes the extension of PROTEAN to include the partitioning method to relieve the state space explosion problem. Instead of modifying the PROTEAN code to include the partitioning method, first it is to partition a Petri Net into subnets, and then to use a new RG module, which was written in MU-Prolog (Naish, 1985), to generate the respective RGS for the subnets. To partition a Petri Net, the user will be asked to nominate the transition to be partitioned. The merits of this paper are the discussions of how complex software can be extended, and of an implementation of a relief strategy, rather than pure descriptions of algorithms as available in the literature.
2 REVIEW OF RELIEF STRATEGIES

There are a number of methods attempting to relieve the state space explosion problem.

— Decomposition is to partition the protocol model into more manageable phases, functions or modules. Valmari (1993) proposed the stubborn set method to reduce state spaces. A stubborn set is a set of transitions which contains at least one enabled. In this method, instead of using all enabled transitions for generating the immediate successor states, a stubborn set is found and only the enabled transitions in the set are used.

— Localisation (Lam and Shankar, 1984) is proposed to avoid the construction of a global reachability tree. Instead of considering an execution of all the processes as a whole and constructing the global tree, localisation considers the executions of all processes separately and constructs N trees with some kind of global information included within each local tree.

— Modification attempts to adapt the global reachability analysis either by using better algorithms (Kakuda et al., 1986) or by adding more assumptions about the protocol (Gouda and Han, 1985). For example, West (1978) made one assumption that two processes progress in equal speeds. Gouda and Han (1985) claimed that such a fair reachability analysis constructed a smaller reachability tree.

— Reduction is based on the generation and execution of the limited set of states only, e.g. the states which assume equal (Rudin and West, 1982) or maximal (Gouda and Yu, 1983) progress speeds of both entities. However, this approach reduces both the complexity and the verification power of RA. A similar effect is achieved in the case of the use of closed cover (Gouda, 1984).

— Simplification of state descriptions is intended to reduce the space needed to store the states and the time needed to compare and analyse these states. For example, Holzmann (1988) proposed a state vector method in which the state of a protocol is given a vector of arbitrary values. Interpreting the state vector as a mere bit vector means interpreting it as a 'number'. This method claimed that in the order of $10^6$-$10^7$ states can be analysed conveniently in minutes of CPU time on a medium size machine.

— State space caching, or on-the-fly verification, is a state space exploration method that stores all the states of just one execution sequence plus as many previously visited states as available memory allows. This approach was first proposed in (Holzmann, 1984) and rediscovered in (Jard and Jerson, 1990). Since then, similar ideas and algorithms have been designed (Godefroid and Wolper, 1992).

— Redundancy is based on the idea that similar execution sequences where only the relative speed of the communicating processes differs can be separately considered. Zafiropulo (1978) introduced a duologue matrix analysis for protocol validation, though it is restricted to modelling interactions of just two non-cyclic processes. Itoh and Ichikawa (1983) extended the work of Zafiropulo to multiprocess systems and introduced the “reduced implementation sequences” to reduce the complexity of the reachability analysis.

3 PARTITIONING METHOD

This section describes the partitioning method (Tridgell, 1987) in detail. Recall that a five tuple Petri net structure can be reduced to a four tuple structure, $N$, by removing the attribute of initial marking, $M_0$, and becoming the following Petri net structure (Peterson, 1981):

$$N = (P, T, I, O), \text{ where}$$

$$P = (P_1, P_2, P_3, P_4), \quad m \geq 0$$

$$T = (t_1, t_2, t_3), \quad n \geq 0$$

$P$ and $T$ are sets of places and transitions respectively such that $P \cap T = 0$.

$I = \text{Input function}$

$O = \text{output function}$

Let us examine the Petri Net $N$ as shown in Figure 1. The partitioning method is designed to divide $N$ into Petri nets $N_1$ and $N_2$.

$$N = (P, T, I, O)$$

$$P = (P_1, P_2, P_3, P_4)$$

$$T = (t_1, t_2, t_3)$$

$M_0$, the initial marking = $(1 \ 0 \ 1 \ 0)$

$I = (p_1, p_2, p_3)$

$O(t_1) = (p_1)$

$I(t_2) = (p_4)$

$O(t_2) = (p_2, p_3)$

$I(t_3) = (p_3)$

$O(t_3) = (p_4)$

Figure 1: Main Petri Net.

To partition a net, we need the following steps:

1. Partitioning a net into subnets is governed by the following restrictions:
   - The boundary or common transition must be selected such that there are arrows going through each transition from one subnet to the other. Failing to satisfy this condition results in deadlock or insignificant subnets.
--- No common place should exist between an input/output arc of a non-boundary transition of a subnet, and between an input/output arc of a non-boundary transition of other subnets. The reachability tree method can be used to check the validity of a partition.

2 Transitions are partitioned such that the sets $T_1$ and $T_2$ are formed as follows:

- $T_1 = \{t_1, t_2, t_3\}$, (Figure 2)
- $T_2 = \{t_2, t_3\}$, (Figure 3)
- $T_{com} = \{t_2, t_3\}$

where $T_1$, $T_2$ are the sets containing the non-boundary transitions, and $T_{com}$ is the set of transitions common to $T_1$ and $T_2$.

3 The input arcs of $N_1$ consist of all input arcs to $T_1$ and the input arcs from $N_1$ to $T_{com}$, and the input arcs of $N_2$ contain all input arcs to $T_2$, and the input arcs from $N_2$ to $T_{com}$. The sets $I_1$ and $I_2$ will then contain the following elements:

- $I_1 = [I(t_1) \cup I(t_2) \cup I(t_3)] = (p_1, p_2, p_3)$
- $I_2 = [I(t_2) \cup I(t_3)] = (p_4)$

An important differentiation is to be drawn between sets and bags. A bag may have more than one occurrence of any element while a set may have just one occurrence. With such a distinction, any bag $I(t_i)$ may contain duplicated places, but $I_1$ and $I_2$ are sets that are entitled to only a single occurrence of the involved places.

4 The output arcs of $N_1$ comprise all output arcs from $T_1$, and the output arcs from $T_{com}$ to $N_1$, and the output arcs of $N_2$ consist of all output arcs from $T_2$, and the output arcs from $T_{com}$ to $N_2$. The sets, $O_1$ and $O_2$, then have the following elements:

- $O_1 = [O(t_1) \cup O(t_2) \cup O(t_3)] = (p_1, p_2, p_3)$
- $O_2 = [O(t_2) \cup O(t_3)] = (p_4)$

The differentiation between bags and sets is to be recalled.

5 $P_1$ and $P_2$, the sets of places in $N_1$ and $N_2$ respectively, consist of the following:

- $P_1 = I_1 \cup O_1 = (p_1, p_2, p_3)$
- $P_2 = I_2 \cup O_2 = (p_4)$

Thus, Petri Net $N$ is divided into subnets $N_1$ and $N_2$.

3.1 Analysing Subnets
After a net has been subdivided, the question of how these subnets are to be analysed arises. Given an original net, $N$, with initial marking $M_0$, the following steps are taken to analyse the subnets:

1 Select any subnet. The marking of each place commences with the value given by the initial marking of the original net.

2 Consider a subnet partitioned through $T_{com}$, use the reachability tree method or the matrix equation approach to analyse the subnet under consideration.

3 Repeat steps 1, 2 for each of the remaining subnets.

The RG for $N$ is shown in Figure 4, that of $N_1$ in Figure 5, and that of $N_2$ in Figure 6. Partitioning this Petri Net into two subnets is just to present the idea of this method. In fact, this method can be applied successfully to a greater number of partitions. If the original net has $m$ places, and $n$ transitions in an original net, an analysis using conventional reachability graph generation methods is of the order $m^n$, but using the partitioning method, the order is on the average $(m/n)^{nl}$ for each subnet, where $l$ is the number of subnets or partitions.
4 PROTEAN

PROTEAN (PROTocol Emulation and ANalysis) (Billington et al., 1988), developed by Telecom Australia Research Laboratories, is a computer-aided tool for verifying communication protocols. It is based on Numerical Petri Nets (NPNs) (Symons, 1978), which are an extension of Petri Nets. With PROTEAN, an NPN specification is the starting point of protocol verification. NPN specifications may be created, edited, stored, combined with other NPNs, structured, listed, displayed, and analysed.

PROTEAN contains several programs to investigate properties of a RG. They include interactive simulation, exhaustive reachability analysis, and several directed graph analysis facilities. RGs can be automatically generated and displayed. Based on a RG, there are facilities that could determine deadlocks and livelocks.

Figure 4: The Reachability Tree for the Main Petri Net.

Figure 5: The Reachability Tree for the First Subnet.

Figure 6: The Reachability Tree for the Second Subnet.
EXTENDING PROTEAN WITH THE PARTITIONING METHOD

The programs are listed below:

NET is the program that reads in files containing NPNs and then generates the RG for them.

GRAPH displays an RG produced by the NET program.

LIVENESS operates on an RG and identifies all the Strongly Connected Components (SCCs) of the graph, that is groups of markings which can be reached starting from and ending at any member of the group.

LANGUAGE reduces an RG to show only the language of specified transitions

SCENARIO takes as its input a sequence of either transitions or markings and outputs a list of sequence from an RG consistent with the input sequence.

PATH determines which nodes in a graph cannot reach a specified node.

4.1 The RG Module

The NET program is the main program of PROTEAN. It is made up of a number of modules, and simply prompts a user to find which module should be called. The modules are Common, Init, Create, Alter, List, Tsort, Outputnpn, Simulate, Drawnpn, NpnLayoutEditor and Printnpn.

Among these modules, the one that is directly related to this work is the Simulate module which performs the analysis of the currently defined NPN. It produces the RG for the NPN, and consists of about 1,600 lines of codes. This module must be run before any of the further analysis programs are run.

The algorithms are as follows:

1 Prompt a user to initialise the net. This is performed by placing requested tokens into places on the net. Global variables can also be initialised with user specified values; otherwise they are assumed to be zero. Let the state of the net at this stage be the initial state of the system, referred to as marking 1. Initialise the Marking List with this entry. Also mark marking 1 as not having been fired yet by placing an entry in the Marking-Not-Fired-Yet-List.

2 REPEAT
   Choose a marking that has not been fired yet.
   Write out this marking.
   For each enabled transition of this marking do
   Write out this transition.
   Fire this transition and generate the resultant marking.
   If this is a new marking then place an entry in the Marking-List and the Marking-Not-Fired-Yet-List.
   UNTIL all possible markings have been fired.

5 IMPLEMENTING THE PARTITIONING METHOD

The NET program generates RG for an NPN specification which is converted to Prolog format using a translation program. This NPN in Prolog format is then partitioned into subnets, and a Prolog RG program will generate RGs for the subnets. An overview of this implementation is shown in Figure 7.

5.1 Format Translation Programs

The NPN specification input in PROTEAN format is stored in a "net" file. In order to execute the Prolog version of the reachability module, the input must be changed to Prolog format which is stored in a "pl" file. The format translation programs are used to convert an NPN specifications in PROTEAN format to MU-Prolog format. These translation programs are written in the C programming language. They are linked to an executable program by the name of "net2pl". They consist of a compiler which reads input net specifications and parses through it to check for any lexical and syntactical errors. If the input specification is free from errors, they will be completely translated into Prolog formats. Then the specifications are executed by a MU-Prolog interpreter. A brief description of the components of this compiler is provided below.

- parser.c: description for NPN parser
- lex.c: lexical analysis module for NPN parser
- string.c: string table for NPN parser
- symbol.c: symbol table for NPN parser
- tree.c: structure tree module for NPN parser
- check.c: static semantics checking module for NPN parser
- find.c: NPN parser module for finding declarations
- type.c: type checker module for NPN parser
- out.c: Prolog code generation module for NPN parser
- printtree.c: structure printing module for NPN parser

The translation works like a normal compiler where characters delimited by blanks are collected as symbols and put together in a table. It then checks for any syntax error, uniqueness of symbols, type error, and declaration error, etc.
If no error is found, the input net specification will be transformed into a parse tree reflecting the logical or semantic relationship of each transition and its associated input and output arcs. The compiler then, according to the parse tree, writes the input net specifications in Prolog format.

5.2 The Partitioning Program
The partitioning module, written in "C" language, partitions a large input net into a number of subnets while retaining the properties of the whole input net. It asks a user to enter the transitions to be partitioned and store the subnets with file names specified by the user. For example, given a net with three transitions, say t₁, t₂ and t₃, to be partitioned, the following steps are to be taken:

— Starting from transition t₁, the program searches for a transition with an input place that matches the output place of t₁.
— If such transition, say t₂, is found and it is neither t₁ nor t₃, then this transition is a member of the subnet associated with t₁, say s₁. The procedures are then repeated with t₂ as starting point and search for its successor until the new transition found is either t₁ or t₃.
— Repeating the last step until all members of s₁ are found.

Members of the second subnet, s₂, can be found by using the same method and applying t₂ as the starting point but this time the process will only stop if the new transition found is either t₁ or t₃. Also, members of the third subnet, s₃, can be found by using t₃ as the starting point, the process terminating when the new transition found is either t₁ or t₃.

This program consists of about 220 lines of C code, implementing the algorithms discussed above and those described in Section 3.

5.3 The Prolog RG Program
After the specifications have been translated into Prolog formats, the next task is to generate the RG and to analyse it. These main programs are written in MU-Prolog. They consist of three parts and are stored in three different files: "simulate.nl", "analyze.nl", and "bag.nl". Given an input net format based on NPNs, the program, "simulate.nl", generates the RG. The program "analyze.nl" analyses the RG to determine if any protocol error, such as deadlock, exists. The program, "bag.nl", provides utility functions to be called by "simulate.nl" and "analyze.nl". These programs are called from the MU-Prolog shell and the output are printed onto standard output. The algorithm to generate the RG is based on the conventional reachability analysis algorithm.

5.3.1 RG Generation Module
Given an input net specification, there are many approaches to generating the RG. One of the approaches is the breadth-first order. This approach is to first visit those nodes that are closed to the initial node. An initial node is a place with an initial token in an input net. This results in a search process that tends to develop more into breadth rather than into depth. Difficulties may arise due to the maintenance of the whole set of alternative candidate paths. Therefore, generate(Verbose) is true if all possible paths are found, given a set of initial nodes.

To do the breadth-first order, we do the following:

**Initialisation**
1. Sort the initial nodes (Rawplaces) to ensure they are unique and put them into a new list (Startplaces).
2. Initialise the total marking count (total) and current marking count (curr) to 1 and 0 respectively.
3. Place one of the start nodes into the list of the reachability set and initialise the marking number to 1 (i.e. equal to total marking number).

Note the total marking count always indicates the total number of marking in the reachability set and the current marking count shows that the child of this marking (i.e. one-step extension) is being generated.

**Generation**
1. Generate the set of all possible one-step extensions of this path, adding this set of extensions at the end of the candidate set and executing a breadth-first order on this updated set.
2. Get the current marking number.
3. If the current marking number is less than or equal to the total marking number, then find all the successors of this marking and put them at the end of the reachability set.
4. Update the total marking count.
5. Repeat (1) and (2) until the current marking number is greater than the total marking number. (i.e. the reachability generation has been completed.)

This Prolog RG program consists of about 186 lines of source code.

5.3.2 Analysis Module
After an RG has been generated, an analysis module is needed to analyse the properties of the net (such as cycle, strong connected components, deadlock, and livelock). At this stage, unlike PROTEAN, the analysis module can only detect deadlock and livelock. The algorithms for detecting deadlock and livelock are based on the "reach" and "link" clause created by the reachability graph generation module. For example, a deadlock is detected if a marking does not have outgoing links to another marking.

5.4 Output Format
This RG is presented in Prolog format and some conversion programs are needed to change the output back to the user-friendly PROTEAN output format. The output in this form is suffixed with " .gr". The output basically consists of two classes of clauses.

The first class of clauses commences with "reach". It describes a marking and the information associated with this marking. The information includes the place name, the token type and name, and the values of P-Variables. The following is an example of a "reach" clause.

reach(mark (users@1("user_info"),[1],[1],"sta0","sta0",true,true,true,[1],1))
The syntax of this clause is:

\[
\text{reach}(\text{mark}(\text{place name} @ \text{No. of tokens} * (\text{"token name"})), \text{list of values in respective } P\text{-variables}, \text{marking number}).
\]

The second class of clauses starts with “link”. It describes the relationship of each marking. It can be illustrated by the following statement:

\[
\text{link}(1, \text{a}_{-}\text{req}, 2)
\]

This means that marking one is connected to marking 2 by a transition, namely “a_{-}req”. The syntax of a link clause is:

\[
\text{link}(\text{marking number}, \text{transition name}, \text{marking number}).
\]

A set of these “link” clauses presents the picture of an RG. It is similar to the idea used in the PROTEAN “.reg” file. At this stage, a set of translation programs written in Prolog are provided to translate this Prolog output format into PROTEAN output format. They are “File.run”, “File.mar”, and “File.reg” files.

6 CONCLUSIONS

This paper has described the extension of PROTEAN to include the partitioning method to address the state space explosion problem. Modifying the PROTEAN RG module Pascal source code to achieve this is a very time consuming and difficult task. Instead, the steps involved are: to translate the NPN specification format to that of MU-Prolog, to partition the net into subnets; and to generate the RGs for the subnets using a new RG module which was written in MU-Prolog.

Prolog is the best known logic programming language. A Prolog program is a set of specifications in formal logic which uses the first-order predicate calculus. The idea of using the representational power of the first-order predicate calculus to express specifications for problem solving is one of the central contributions of Prolog. This explains why the Prolog RG module only contains 186 lines of code, as compared to 1,600 lines of Pascal code for doing the same thing. The time taken to write, debug and test the Prolog module was about two months; whereas it took Telecom Australia a few years to develop the whole PROTEAN software. In conclusion, it is relatively easier to use Prolog to implement an algorithm than a conventional programming language like Pascal, and the approach to extending software is less time and effort consuming, by not modifying existing code but by writing new modules using Prolog.

ACKNOWLEDGEMENT

The use of PROTEAN made available by Telecom Australia, and the assistance of J Pritchard and CW Lee in this work are hereby acknowledged.

REFERENCES


BIOGRAPHICAL NOTES

Richard Lai holds a BE (Hons) and a MEngSc from the University of New South Wales, and a PhD from La Trobe University, Australia. He is at present a senior lecturer and the Director of Undergraduate and Diploma Studies in the School of Computer Science and Computer Engineering at La Trobe University. Prior to joining the University in 1989, he has worked in the computer and communication industry for about 12 years. He has been a consultant to international companies and has authored about 60 technical papers. He was a member of the international advisory board for the 1995 IEEE Singapore International Conference on Networks and International Conference on Information Engineering, and was on the program committee of the 15th International Symposium on Protocol Specification, Testing and Verification, 1995. His research interests include protocol engineering, computer networking and performance evaluation.
Genetic Algorithm for Fine-Tuning Fuzzy Rules for the Cart-Pole Balancing System

GD Finn
Queensland University of Technology
Brisbane, Australia 4000

Genetic algorithms are used to fine-tune the membership functions and, especially, the large number of fuzzy rules for controlling a cart-pole dynamical system where a hinged pole is supposed to balance on a powered cart confined to a finite length of straight track. The fitness function is determined by the total number of steps to failure of computer simulations (up to a maximum number identified with success) for various initial configurations. Starting from a reasonably knowledgeable set of rules the genetic algorithm efficiently leads to a superior set which provides balance in both bang bang and continuous force modes for a range of update times. It is possible to start with a random set of rules but the derivation of a good set takes much longer. It was found possible to exert influence on the quality of balance by subtracting a penalty from the fitness function for simulations which only provide noticeably unsteady balance. As well, it is shown that a single set of fuzzy rules can be used to achieve a state of balance at essentially arbitrary positions of the cart on the track.

1 INTRODUCTION
In the framework of a dynamical control problem we investigate the usefulness of genetic algorithms to derive a superior set of fuzzy rules starting from an inferior set or even a random set. The two-dimensional control experiment in which a hinged pole balances on top of a movable cart confined to a finite length of straight track provides a suitable problem since it involves four state variables and accordingly has hundreds of fuzzy rules. Further, it is not a totally unfamiliar environment to most humans so that a person using heuristics, as we have done, may eventually compile a fair working set of rules, especially with access to a computer model for testing by simulation. As well, the cart-pole is a benchmark for control systems and the quality of the fuzzy logic controllers we obtain competes with controllers designed by other methods. Indeed, our fuzzy controllers for the cart-pole system behave very well over a considerable range of conditions and it would appear that only the "linear controller" with weights obtained by pure random search is better as well as being more easily designed (Geva and Sitte, 1993a). The present investigation indicates that the technique of obtaining rules and fuzzy set membership functions by genetic algorithms should be widely useful for problems having in the order of hundreds of fuzzy rules.

Fuzzy set theory was formally introduced by Zadeh (1965), and subsequently expanded and explicated at length by a number of authors, for example Kosko (1991). Fuzzy logic controllers have been successfully applied to control numerous physical processes and are embedded in a number of industrial and household products. For many applications the compilation of fuzzy rules can limit the practicality of a fuzzy control system unless an effective automated learning algorithm is available. Karr (1992) reports an adaptive system for control of the pH of a chemical process introducing a learning component that employs a genetic algorithm to find new rules appropriate to a changing environment. In the control of an anaesthetic drug, Linkens (1992) has also employed a learning algorithm based on a genetic algorithm to determine the relevant rules. Berenji (1992) employs a modified adaptive heuristic critic neural network approach to fine-tune fuzzy rules with his GARIC (generalised approximate reasoning-based intelligent control) architecture. Jang (1992) uses a methodology he called Temporal Back Propagation to learn fuzzy rules and gives some results for the cart-pole problem. Genetic algorithms can be used to provide an efficient search technique and form a class of learning techniques developed by Holland (1975) and elaborately explained by Goldberg (1989). In this paper we develop their use to specify fuzzy logic controllers for balancing the cart-pole dynamical system. We show that the use of genetic algorithms provides a practical method for finding the large number of fuzzy rules and the membership functions of the control and the four state variables starting from reasonable working sets.

After a short discussion of genetic algorithms and their use in the present context in Section 2, we discuss the cart-pole
functions though the number of categories for each variable is selected manually by the designer.

An algorithm is used to determine suitable triangular membership rules. A separate algorithm with the flavour of a genetic controller with a genetic algorithm to learn a suitable set of fuzzy control systems. Specifically, we employ a fuzzy logic controller's inability to decide whether any particular decision is beneficial or detrimental to the overall goal of obtaining balance the control problem is a difficult one. On the other hand, it is also quite straightforward to realise a system in which at each time step knows which of the available control decisions is optimal. It can also take the form of a less knowledgeable critic which can only assess how well the controller fares overall. In the case of balancing a pole the critic can assess the effectiveness of a set of rules by measuring the time to failure when the pole falls over or otherwise exceeds the limits imposed on the experiment. The nature of genetic algorithms is explained in more detail in Section 4 in the context of compiling fuzzy rules for the cart-pole system.

2 FUZZY LOGIC AND GENETIC ALGORITHMS

The shape of fuzzy membership functions is generally discretionary provided the membership functions of adjacent categories overlap sufficiently. However, the specifications of the number of categories and their ranges needs attention as, of course, does the selection of a good set of fuzzy rules.

In the case of learning with a genetic algorithm the fuzzy rules are encoded as gene strings. Starting from an initial gene string others are produced using the techniques of reproduction, crossover, and mutation as explained by Goldberg (1989). The gene strings in a population are assessed for fitness with strings producing good performance strongly participating in the formation of the next generation. Depending on the worth of the assessment algorithm gene strings that perform badly tend to be replaced in favour of those that perform well.

Assessment of behaviour of plant dynamics under a set of fuzzy rules may, if a model is available, be simulated on a computer. The assessment may take the form of a teacher which at each time step knows which of the available control decisions is optimal. It can also take the form of a less knowledgeable critic which can only assess how well the controller fares overall. In the case of balancing a pole the critic can assess the effectiveness of a set of rules by measuring the time to failure when the pole falls over or otherwise exceeds the limits imposed on the experiment. The nature of genetic algorithms is explained in more detail in Section 4 in the context of compiling fuzzy rules for the cart-pole system.

3 THE CART-POLE DYNAMICAL SYSTEM

The control problem of balancing a cart-pole system is a benchmark for control task learning problems. Numerous attempts at control by a plethora of methods are documented in the literature, see for example Michie and Chambers (1968); Barto, Sutton and Anderson (1983); Widrow and Smith (1963) and Geva and Sitte (1993a). Due largely to the controller's inability to decide whether any particular decision is beneficial or detrimental to the overall goal of obtaining balance the control problem is a difficult one. On the other hand, Geva and Sitte (1993a) show that good linear controllers for the cart-pole are easily obtained by random search and on this basis it is also quite straightforward to realise a system in hardware under software control. However, we are interested in investigating a general method that can be applied to other control systems. Specifically, we employ a fuzzy logic controller with a genetic algorithm to learn a suitable set of fuzzy rules. A separate algorithm with the flavour of a genetic algorithm is used to determine suitable triangular membership functions though the number of categories for each variable is selected manually by the designer.

In the cart-pole problem a pole stands atop a cart and is constrained by a hinge (assumed here to be frictionless) to move in a vertical plane. The cart moves along a straight track of fixed length between two barriers. At every time step, a decision is made to deliver a force to the cart either to the left or the right. The force is assumed to act unvaryingly for the duration of a time step. The aim of the experiment is to balance the pole on the cart; failure occurs if either the pole falls over or the cart hits the barriers. The system is inherently unstable and humans find it difficult to predict a sequence of decisions that achieves balance over a significant period of time. However, a human can after a little thought or experimentation distil some useful decision-making inferences from which a useful base set of fuzzy rules can be designed. Consider the case where the cart is momentarily stationary in the middle of the track with the pole leaning to the right with zero angular velocity. The application of a force to the right kicks the cart to the right (pushing it under the pole so to speak) and tends to correct the lean of the pole. Even in more difficult situations where, for example, the cart is near the right end of the track and moving to the right, if the pole leans to the right a force should be applied to the right. The imperative is to cause the pole to correct itself and perhaps lean somewhat to the left; then a force to the left can be applied to correct the cart's movement to the right. Thus, the correct overall strategy is to attend to the lean of the pole first.

Our cart-pole experiments operate in either continuous mode when the force delivered at any time step can have any value up to a set maximum, or in bang bang mode when a force of constant magnitude must be applied to the left or the right. In either mode, computer simulation of the movement of the cart-pole system is relatively straightforward requiring numerical integration over time of the dynamical differential equations of motion (Barto, Sutton and Anderson, 1983). We have also adopted standard values for the masses of the pole and cart as well as for the lengths of the pole and track as presented in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track length</td>
<td>4.8m</td>
</tr>
<tr>
<td>Length of pole</td>
<td>1m</td>
</tr>
<tr>
<td>Mass of cart</td>
<td>1.0kg</td>
</tr>
<tr>
<td>Mass of pole</td>
<td>0.1kg</td>
</tr>
<tr>
<td>(Maximum) magnitude of controlling force</td>
<td>10N</td>
</tr>
</tbody>
</table>

The state of the cart-pole system at any time is completely specified by four state variables — the position and velocity of the cart on the track (x and \( \dot{x} \)) and the pole's angle and angular velocity with respect to the vertical (\( \theta \) and \( \dot{\theta} \)).

Balancing the cart-pole system may have a number of goal states of varying difficulty. For example:

- The system succeeds if the pole does not fall over (or does not exceed a prescribed angle) and the cart does not reach the end of the track. This is a weak goal since the cart-pole system may never be effectively balanced in the static sense with the cart moving around the track and the pole
rarely being stationary and vertical. Thus, the system does not fail nor does it achieve steady balance.

The system is required to become well-balanced, so as to stabilise with the pole static in the vertical position and the cart stationary at a prescribed position on the track, usually the centre. It is certainly not possible to achieve this state perfectly in bang bang mode since zero force is not allowed. In either mode we regard the system as being well-balanced when near static balance is maintained.

Starting the system from different initial configurations allows investigation of those regions of four-dimensional state hyperspace \((x, \dot{x}, \theta, \dot{\theta})\) from which balance is possible. Other control experiments can be designed by altering the update time between decisions. The longer the update time the longer is the controller committed to the application of a particular force so that balance becomes more difficult. We consider problems with update times ranging from 0.02s to 0.05s, that is for controllers running from 50Hz down to 20Hz.

The dynamical equations show that the cart-pole is intrinsically a non-linear system as the applied force \(F\) is a highly non-linear function of the state variables despite the ease of finding robust linear controllers by random search. For linear controllers decisions for the applied force are made from the empirical equation:

\[
F = \alpha x + \beta \dot{x} + \gamma \theta + \delta \dot{\theta}
\]

where \(\alpha, \beta, \gamma\) and \(\delta\) are constants. The three-dimensional hyperplane of equation (1) divides the four-dimensional state space into two regions according to the sign of the right side of equation (1). As Geva and Sitte (1993a) explain, the state classification problem resulting from the non-linear dynamic equations can appear linear because the system is essentially very forgiving even in bang bang mode.

Numerical techniques for specifying control effectively quantise the four-dimensional state space into a grid of small cells where a decision (left or right) is associated with each cell. One can visualise the evolution of the dynamical system under some sequence of left-right decisions as the passage of a point in hyperspace. After a force is applied during a time step the point may or may not have changed cells. The cells fall into three categories:

A. The point enters a cell from which failure to balance is inevitable. That is, the system can only progress to another cell of this category.

B. The point enters a cell for which the specific left-right decision is crucial else ultimate failure is unavoidable. That is, the point progresses from a cell of this category directly to a cell of category A unless the correct decision regarding the force is made.

C. The point enters a cell for which the specific left-right decision is not crucial. That is, regardless of the decision the point does not progress to a cell of category A. Rather, the point may progress through a number of cells of category C before entering a cell of category B for which the correct decision is crucial. The quality of the balance achieved can be influenced by decisions made in some of the cells of category C. It is desirable for a controller to make those decisions for category C cells that cause the system to become well-balanced; that is, that effect passage to a cell of class B or C closer to the state of steady balance where \(\theta = \dot{\theta} = 0\).

It is apparent that the decision surface given by equation (1) should endeavour to classify correctly cells belonging to category B. In reality, decision surfaces may well do somewhat less than this. Since we are not aware of the actual distribution of cells among the categories we may well be impressed with a controller even though it may make the wrong decision in a category B cell and therefore eventually fail. Such a cell is perceived to belong to category A for the actual controller whereas for a better controller it would belong to category B. The situation becomes more difficult if a single controller is used for a range of update frequencies since a cell may belong to different categories for different update frequencies.

The division of state space and the classification of cells applies directly to the neural controller of Barto et al (1983) and to the BOXES controller of Michie and Chambers (1968). For fuzzy controllers, state space is divided instead into broad overlapping cells. Fuzzy rules that fire at some stage may be thought of as active cells with the (crisp) state of the system belonging to varying extents to a number of adjacent active cells. If a system ultimately becomes fairly well-balanced in the centre of the track then activity becomes confined to the neighbourhood of the central cell with all state variables being roughly zero.

In order to test fuzzy controllers over a spectrum of starting conditions we consider the following initial configurations:

- **Problem 1:**
  \[
  \theta = 10 \text{ deg}; \quad \dot{\theta} = 10 \text{ deg/s}; \quad x = 0.0 \text{ m}; \quad \dot{x} = 0.0 \text{ m/s}.
  \]

- **Problem 2:**
  \[
  \theta = 0 \text{ deg}; \quad \dot{\theta} = 0 \text{ deg/s}; \quad x = -1.25 \text{ m}; \quad \dot{x} = -1.5 \text{ m/s}.
  \]

- **Problem 3:**
  \[
  \theta = 30 \text{ deg}; \quad \dot{\theta} = 30 \text{ deg/s}; \quad x = -2.0 \text{ m}; \quad \dot{x} = 0.0 \text{ m/s}.
  \]

In problem 1 the system starts with the cart stationary at the centre of the track and with the pole leaning at a small angle to the right with a small angular velocity to the right increasing the lean. Initially the responses of the controller should be to apply force to the right until the lean to the right is corrected or at least until the angular velocity to the left becomes large. In problem 2 the system starts with a vertical non-rotating pole and with the cart on the left side of the track and travelling relatively swiftly to the left. It is incorrect to

---

1 It is possible that for some crisp points in a cell one decision is crucial and for others the other decision is necessary. Such cells would place a fundamental limit on the usefulness of a controller. However, for fuzzy controllers this may be mitigated by the ability of a crisp point to belong to a multiplicity of fuzzy cells. It was not apparent to us that such cells exist in our better controllers running at frequencies \(>20\text{Hz}\).
The
AUSTRALIAN COMPUTER JOURNAL
VOLUME 28
February 1996 to November 1996

CONTENTS

Contributions

1. February, 1996

Harvey, N. and Morris, J. NL: A Parallel Programming Visual Language 2-12
Martin, A. and Chan, M. Information Systems Project Redefinition in New Zealand: Will We Ever Learn? 27-40

2. May, 1996

Ghosh, A. and Crowcroft, J Some Lessons Learned from Various ALF and ILP Applications 42-47
Braun, T., Chrisment, I., Diot, C., Gagnon, F. and Gautier, L. ALF/ILP Based Automated Implementation of Distributed Applications 48-54
Fry, M. and Ghosh, A. Some Issues for Dynamic Synthesis of ALF/ILP Systems 61-65

3. August, 1996

Halpin, T. and Ritson, P. Entity Integrity Revisited 73-80
Levin, L.S. and MacLeod, I.M. Understanding Faults: A First Step in Developing Fault-Tolerant Computer Systems 81-87
Fry, M.R. Management of the IT Industry: A Survey of Critical Challenges 88-95
Renouf, D.W. and Henderson-Sellers, B. Towards a Role-Based Framework Approach for User Interfaces 96-106

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pullan, W.</td>
<td>A Direct Search Method Applied to a Molecular Structure Problem</td>
<td>113-120</td>
</tr>
<tr>
<td>Lai, R.</td>
<td>Extending PROTEAN with the Partitioning Method</td>
<td>121-127</td>
</tr>
<tr>
<td>Finn, G.D.</td>
<td>Genetic Algorithm for Fine-Tuning Fuzzy Rules for the Cart-Pole Balancing System</td>
<td>128-137</td>
</tr>
</tbody>
</table>

**Miscellaneous**

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest Editorial</td>
<td>1, 41</td>
</tr>
<tr>
<td>Errata</td>
<td>71</td>
</tr>
<tr>
<td>Book Reviews</td>
<td>72, 107</td>
</tr>
<tr>
<td>Call for Papers — Multimedia Storage and Archiving Systems</td>
<td>120</td>
</tr>
<tr>
<td>Title</td>
<td>Pages</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>A Comparison of Automatic Protocol Generation Techniques</td>
<td>55-60</td>
</tr>
<tr>
<td>A Direct Search Method Applied to a Molecular Structure Problem</td>
<td>113-120</td>
</tr>
<tr>
<td>A Hacker's Guide to Project Management</td>
<td>109</td>
</tr>
<tr>
<td>A Knowledge-Based Approach to Program Understanding</td>
<td>112</td>
</tr>
<tr>
<td>ALF/ILP Based Automated Implementation of Distributed Applications</td>
<td>48-54</td>
</tr>
<tr>
<td>Artificial Neural Networks for Image Understanding</td>
<td>72</td>
</tr>
<tr>
<td>Computations with Markov Chains</td>
<td>72</td>
</tr>
<tr>
<td>Co-operative Computer-Aided Authoring and Learning</td>
<td>110</td>
</tr>
<tr>
<td>Developing Visual Basic 4 Communications Applications</td>
<td>111</td>
</tr>
<tr>
<td>EasyDesktop 6.0 for Windows 95</td>
<td>112</td>
</tr>
<tr>
<td>E-mail: A Practical Guide</td>
<td>107</td>
</tr>
<tr>
<td>Embedded Systems Programming in C and Assembly</td>
<td>107</td>
</tr>
<tr>
<td>Entity Integrity Revisited</td>
<td>73-80</td>
</tr>
<tr>
<td>Extending PROTEAN with the Partitioning Method</td>
<td>121-127</td>
</tr>
<tr>
<td>Fundamentals of Computer Numerical Analysis</td>
<td>108-109</td>
</tr>
<tr>
<td>Genetic Algorithm for Fine-Tuning Fuzzy Rules for the Cart-Pole Balancing System</td>
<td>128-137</td>
</tr>
<tr>
<td>Global Interface Design</td>
<td>108</td>
</tr>
<tr>
<td>Information Systems Project Redefinition in New Zealand: Will We Ever Learn?</td>
<td>27-40</td>
</tr>
<tr>
<td>Lotus Notes: A Practical Guide</td>
<td>108</td>
</tr>
<tr>
<td>Management of the IT Industry: A Survey of Critical Challenges</td>
<td>88-95</td>
</tr>
<tr>
<td>Microcomputer Interfacing and Applications (2nd Edition)</td>
<td>110</td>
</tr>
<tr>
<td>Multimedia and Hypertext: The Internet and Beyond</td>
<td>72</td>
</tr>
<tr>
<td>Netscape Navigator: Surfing the Web and Exploring the Internet</td>
<td>110</td>
</tr>
<tr>
<td>NL: A Parallel Programming Visual Language</td>
<td>2-12</td>
</tr>
<tr>
<td>Object Solutions, Managing the Object-Oriented Project</td>
<td>107</td>
</tr>
<tr>
<td>Some Issues for Dynamic Synthesis of ALF/ILP Systems</td>
<td>61-65</td>
</tr>
<tr>
<td>Some Lessons Learned from Various ALF and ILP Applications</td>
<td>42-47</td>
</tr>
<tr>
<td>Structured Graphs: A Visual Formalisation for Scalable Graph Based CASE Tools</td>
<td>13-26</td>
</tr>
<tr>
<td>The Coriolis Group, Zen of Windows 95 Programming</td>
<td>111-112</td>
</tr>
<tr>
<td>The Interaction of Compilation Technology and Computer Architecture</td>
<td>109</td>
</tr>
<tr>
<td>The Power of List Ranking on a Reconfigurable Array of Processors with Wider Bus Networks</td>
<td>138-148</td>
</tr>
<tr>
<td>Towards a Role-Based Framework Approach for User Interfaces</td>
<td>96-106</td>
</tr>
<tr>
<td>Towards Predictable ILP Performance — Controlling Communication Buffer Cache Effects</td>
<td>66-71</td>
</tr>
<tr>
<td>Understanding Faults: A First Step in Developing Fault-Tolerant Computer Systems</td>
<td>81-87</td>
</tr>
<tr>
<td>Virus Detection and Elimination</td>
<td>111</td>
</tr>
<tr>
<td>Visual Basic for Windows 95 Insider</td>
<td>111</td>
</tr>
<tr>
<td>Contributor</td>
<td>Pages</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>Ahlgren, B.</td>
<td>66-71</td>
</tr>
<tr>
<td>Antoniou, G.</td>
<td>112</td>
</tr>
<tr>
<td>Bell, P.</td>
<td>107</td>
</tr>
<tr>
<td>Bloesch, A.</td>
<td>72</td>
</tr>
<tr>
<td>Björkman, M.</td>
<td>66-71</td>
</tr>
<tr>
<td>Braun, T.</td>
<td>48-54</td>
</tr>
<tr>
<td>Chan, M.</td>
<td>27-40</td>
</tr>
<tr>
<td>Chrisment, I.</td>
<td>48-50</td>
</tr>
<tr>
<td>Clark, S.</td>
<td>111</td>
</tr>
<tr>
<td>Coombes, J.</td>
<td>108</td>
</tr>
<tr>
<td>Crowcroft, J.</td>
<td>42-47</td>
</tr>
<tr>
<td>Dairaine, L.</td>
<td>55-60</td>
</tr>
<tr>
<td>De Silva, R.</td>
<td>55-60</td>
</tr>
<tr>
<td>Diot, C.</td>
<td>48-50</td>
</tr>
<tr>
<td>Finn, G.D.</td>
<td>128-137</td>
</tr>
<tr>
<td>Fry, M.</td>
<td>55-60, 61-65, 88-95</td>
</tr>
<tr>
<td>Fulcher, J.</td>
<td>72</td>
</tr>
<tr>
<td>Gautier, L.</td>
<td>48-50</td>
</tr>
<tr>
<td>Gore, A.</td>
<td>108-109</td>
</tr>
<tr>
<td>Gunningberg, P.</td>
<td>66-71</td>
</tr>
</tbody>
</table>
immediately arrest the leftward movement of the cart before it reaches the left barrier by applying force to the right since this causes the pole to fall to the left which, in turn, requires correction by applying force to the left when the cart is still closer to the left barrier. Rather, the first imperative is to apply force to the right until the pole is falling sufficiently rapidly to the right even though this causes the cart’s velocity to the left to increase. Later, when the system has been brought under temporary control, application of force to the right both corrects the rightward fall of the pole and slows the leftward motion of the cart. Problem 3 represents an even more difficult initial configuration since the pole initially leans well to the right of the vertical while rotating to the right. As for problem 1 initially force must be applied to the right until the pole has a substantial rotation to the left. These strategies translate easily into empirical fuzzy rules.

We adopt the following linguistic categories with their usual abbreviations: NL and PL (negative and positive large), NM and PM (negative and positive medium), NS and PS (negative and positive small), and ZE (zero). We have employed the linguistic categories NL and PL only for \( \theta \) giving it seven categories since \( \theta \) can vary widely. All other variables including the control force \( F \), need only the five categories NM, NS, ZE, PS, and PM. We have listed these symbols together with their meanings and other abbreviations we use for them in Table 2.

In the case of bang bang controllers crisp values for the decision forces are calculated (using the standard centroid method) only for their sign, while, for continuous mode, \( F \) is equal to the computed value (or perhaps proportional to it).\(^2\)

For the purpose of presentation in the figures it is desirable to have a single character for the linguistic categories. We adopt – for NM, n for NS, o for ZE, p for PS, and + for PM — see Table 2.

After some experimentation we were able to compile a set of fuzzy rules and a set of membership functions empirically to give a fuzzy controller that maintains reasonable balance for problems 1-3 when operating at 50 Hz in both bang bang and continuous modes though not at the lower frequencies which are more difficult to control. The empirical membership functions are shown as solid lines in Figure 1 while in Figure 2 and our later figures, the fuzzy rules are presented in the one-character code as specified in Table 2. The figure should be interpreted as fuzzy values in a four-dimensional array corresponding to the four state variables, organised as 25 blocks each containing 5 rows and 7 columns, the blocks themselves lying in a 5 by 5 array. Within a block successive positions along a row from left to right show the consequents of rules \( F \) for linguistic categories of \( \theta \) from NL to PL respectively. Successive positions down a column from top to bottom show \( F \) for the linguistic categories of \( \theta \) from NM to PM. The blocks themselves correspond to linguistic categories of \( x \) and \( \dot{x} \) as labelled in Figure 2. As an example consider the + in the top left corner of Figure 2. This represents the rule:

\[
- \text{To the extent that } x \text{ is NM and } x \text{ is NM and } \dot{x} \text{ is NM and } \dot{x} \text{ is NL then } F \text{ is } + (\text{that is PM}).^3
\]

One of the characteristics of fuzzy sets is that a crisp value may belong to more than one fuzzy category. We see from Figure 1 that, for the degree of overlap of the fuzzy categories we employ, it is possible for a crisp value of any variable to belong to up to two fuzzy sets. Thus it is possible for a crisp value of the state of the system \((x, \dot{x}, \theta, \dot{\theta})\) to belong to some extent to up to 16 of the 4-dimensional fuzzy cells.

---

\(^2\) For problem 3 in continuous mode \( F \) needs to be multiplied by a constant in the order of two with the resultant value of \( F \) clipped as necessary to the maximum control force in order to obtain a good controller.

\(^3\) Following general custom we assume the consequent \( F \) belongs to its fuzzy category (in this case PM) to the minimum of the extents that the four antecedent variables belong to their categories.

---

**Table 2 - List of symbols used: The symbols NM to PM appear in the figures using the single character code listed in the third column.**

<table>
<thead>
<tr>
<th>linguistic meaning</th>
<th>usual symbol</th>
<th>single character</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative large</td>
<td>NL</td>
<td>NL</td>
</tr>
<tr>
<td>negative medium</td>
<td>NM</td>
<td>n</td>
</tr>
<tr>
<td>negative small</td>
<td>NS</td>
<td>o</td>
</tr>
<tr>
<td>zero</td>
<td>ZE</td>
<td></td>
</tr>
<tr>
<td>positive small</td>
<td>PS</td>
<td>p</td>
</tr>
<tr>
<td>positive medium</td>
<td>PM</td>
<td>+</td>
</tr>
<tr>
<td>positive large</td>
<td>PL</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 1: Membership functions for \( \theta \), \( \dot{\theta} \), \( x \), \( \dot{x} \), and \( F \). Solid lines denote empirical membership functions; broken lines denote genetically altered membership functions. The categories from the left in each case are ZE, PS, and PM (as well as PL for \( \theta \)). Note in all cases only the half of category ZE on the positive side of the axes are shown.**
GENETIC ALGORITHM FOR FINE-TUNING FUZZY RULES FOR THE CART-POLE BALANCING SYSTEM

<table>
<thead>
<tr>
<th>x</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
</tr>
<tr>
<td>+++)-- -nnnn -nnnn ----nn -nnnn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+----+ -nnno -nnno ----oo -nno0+p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NM</td>
<td>nnnnnp nnnnpp nnnopp nnnopp no+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>++++++ nopp++ nopp++ nopp++ op+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-++++n -++++n -++++n -++++n -++++n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-++++n -++++n -++++n -++++n -++++n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS</td>
<td>+nnpp+ +nnpp+ nopp++ nopp++ p+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n+++pp+ nopp++ nopp++ p+++++ op+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZE</td>
<td>-nnnn+ -nopp+ -nopp+ nopp++ -nopp++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nnnnn+ -nopp+ nopp++ oo+++++ np+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nnnnn+ nopp++ nopp++ pp+++++ np+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>+nnpp+ +nnpp+ nopp++ nopp++ pp+++++ pp+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nopp++ +nnpp+ +nnpp+ pp+++++ pp+++++</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----n ----n ----n ----n ----n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>-nnpp+ -nnpp+ -nnpp+ -nnpp+ -nnpp+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nopp++ +nnpp+ +nnpp+ +nnpp+ +nnpp+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nopp++ +nnpp+ +nnpp+ +nnpp+ +nnpp+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Empirically obtained rules tested on the 3 problems at 50 Hz — useful for problems at 50 Hz only.

4 CODING FUZZY RULES FOR A GENETIC ALGORITHM

In order to employ a genetic algorithm, a set of fuzzy rules (such as those of Figure 2) has to be encoded as a gene string. We use the position in a gene string to reflect the categories of the four state variables (just as the position of a character in Figure 2 does), and the value in the gene string to code the linguistic category of the force \( F \). An obvious antisymmetry exists in the cart-pole system since if a force \( F \) is desirable for state \((x, \dot{x}, \theta, \dot{\theta})\), then the opposite force \(-F\), is desirable for the state \((-x, -\dot{x}, -\theta, -\dot{\theta})\). The fuzzy membership functions should reflect this antisymmetry. Hence, out of the 875 cells in our formulation only 438 rules are independent and should be coded in a gene string. The position in the gene string assigned to the 4-dimensional fuzzy cells would appear to be arbitrary. However, it is convenient to have a simple mapping from position in figures like Figure 2 to position in the gene string and we briefly recount our assignment for the 438 rules which is indicated in Figure 3. We see from this figure that the first 35 positions in the gene string correspond to the categories for which \((x, \dot{x}) = (NS, NL)\). As an example of the coding, consider the category \((x, \dot{x}) = (NS, NM)\) of the particular gene string shown in tabular form in Figure 2. We adopted the column-major policy of mapping to a gene string so that each of the seven columns from the left are placed in the string in turn to produce the first 35 symbols in the gene string as shown below in the single character code listed in Table 2:

\[-nnp+n+ -nop - -nop + -noppnn+op+++.\]

We emphasise that each symbol in the string represents the consequent force. Thus the first symbol \(='-\) in the gene string specifies that for the fuzzy cell \((x, \dot{x}, \theta, \dot{\theta}) = (NS, NM, NL)\) the force \(F\) to be applied is \(='-\), that is \(NM\) or “negative medium”.

The next 35 positions in the gene string (positions 36 to 70) correspond in a similar fashion to the category \((x, \dot{x}) = (ZE, NM)\) as indicated in Figure 3. The coding proceeds in this manner with complete \((x, \dot{x})\) blocks being added to the string in turn until the last complete block \((x, \dot{x}) = (PM, PM)\) is placed in positions 386 to 420. The complete blocks marked in Figure 3 with an ‘A’ are not coded into the gene string since they can be derived from coded blocks by consideration of the antisymmetry of the cart-pole system. Finally, that part of the central block corresponding to \((x, \dot{x}) = (ZE, ZE)\) in Figure 3, that corresponds to independent rules, is added to form the extreme right end of the gene string in positions 421 to 438. The final 438th element of the string corresponds to the fuzzy cell \((x, \dot{x}, \theta, \dot{\theta}) = (ZE, ZE, ZE, ZE)\). A coded gene string can automatically and straightforwardly be identified with and converted to a set of fuzzy rules, one rule for each position in the gene string. Conversely, a set of fuzzy rules can be represented by a gene string. The membership functions shown in Figure 1 are not coded in the gene string representation above but may themselves be adapted by a separate genetic algorithm and gene string representation.

It is crucial in a genetic algorithm that a set of fuzzy rules (as represented by a gene string) be evaluated for fitness for the task for which they are designed. In the present context we evaluate the performance of the set of rules defined by the gene string (with a set of pre-determined membership functions such as those of Figure 1) by how well the cart-pole dynamical system behaves under their use. We set up a list of dynamical tasks in the form of the cart-pole system initialised by specifying crisp values for their initial states \((x, \dot{x}, \theta, \dot{\theta})\). Then the fitness of gene string is determined by the performance of the fuzzy controller appropriate to the set of fuzzy rules. This can be quantified by the number of time steps the system takes to failure or, after a specified amount of simulated time has passed, when it is deemed to have succeeded and the fitness score assigned a maximum value. Since the

---

4 For actual implementation these single character codes may be conveniently further translated to digits, say, from 1-5.

5 Arguably this last element could be omitted since symmetry considerations dictate that the force \(F\) applicable to this cell should be \(ZE\). However, we included it in our study.
dynamic system is Markovian, any crisp value taken on by the

cart-pole may be regarded as a potential initial state. Thus the
evaluation of fitness by employing simulation with the initial
configurations specified in Problems 1-3 actually produces
robust controllers capable of controlling a wide variety of
starting configurations.

<table>
<thead>
<tr>
<th>( x )</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>ZE</td>
<td>36</td>
<td>141</td>
<td>421</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>175</td>
<td>435</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>71</td>
<td>176</td>
<td>246</td>
<td>361</td>
<td>A</td>
</tr>
<tr>
<td>105</td>
<td>210</td>
<td>280</td>
<td>385</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>106</td>
<td>211</td>
<td>216</td>
<td>316</td>
<td>386</td>
</tr>
<tr>
<td>140</td>
<td>245</td>
<td>315</td>
<td>350</td>
<td>420</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: The positions of the consequents to the various rules in the gene
string are indicated. Areas marked with an “A” denote regions for which
the rules are inferred by antisymmetry.

5 GENETIC ALGORITHM FOR FUZZY RULES

Genetic algorithms operate by a simplified programmatic
simulation in the spirit of biological gene strings. A current set
of gene strings forms the present generation. Genes strings are
selected for reproduction based on their suitability as mea­sured by a fitness score based on some appropriate criterion.
Frequently, the fitness score is a single number and the
probability of selecting a gene string to be involved in repro­duction, and to become a parent of gene strings in the next
generation, is directly proportional to its fitness score. The
intention of the genetic algorithm is to attempt to produce gene
strings in successive generations with generally increasingly
high fitness scores by a process that is more efficient than
simple unguided random selection. To this end the notion of
pairs of biological parents is simulated whereby two derived
gene strings are produced from two parents by simple modi­
fications of the parental gene strings akin to processes in
nature. The hope is that by choosing parents of relatively high
fitness some children of even higher fitness will be created for
the next generation. In order to implement algorithms in the
spirit of the biological context for determining the gene strings
in successive generations, random selection is employed
extensively, first to select a parent or parents from among the
population of the present generation and then to implement
modifying processes to produce related gene strings for the
next generation.

At the commencement of each reproductive step the

genetic operator of crossover is selected based on a user­specified probability of occurrence \( p_c \). Otherwise mutation
is selected at random. If crossover is selected a pair of
parent gene strings from the current generation is selected
by a simulated roulette wheel whereby each parent is
selected with a probability proportional to its fitness score.
A single cross-over position \( Y \) is selected uniformly and at
random in the parental gene strings for a crossover to be
engineered.\(^6\) The single crossover position is marked in
each string and the two parent strings are cut there. Two
new “offspring” gene strings \( X \) and \( Y \) say, are generated for
the next generation. The gene string for \( X \) is a copy of the
head (that is, the part of the string to the left of the\(^7\)
crossover position) of the gene string of one parent
concatenated with a copy of the tail (that is, the part of the
string to the right of the crossover position) of the other
parent. \( Y \) is formed from a copy of the head of the other
parent concatenated with the tail of the one. The offspring
gene strings then represent two new (but somewhat related)
instances of the fuzzy rule set with the 438 independent fuzzy rules coded into 438 genes of each of the
two offspring gene strings.

Normally, the probability of mutation is quite small rela­tive to the probability of crossover since the purpose of
mutation is to occasionally add fresh genes to a pool that might
otherwise be permanently absent.\(^7\) Should mutation have
been selected at the start of each reproductive cycle then a
single parent gene string is selected at random from the parent
generation and the number of mutations in the string is
selected at random based on a user-specified probability of
occurrence of mutation per gene \( p_m \). Many genes in a gene
string correspond to extreme fuzzy cells that never fire in any
simulation that avoids failure regardless of the coding of the fuzzy rule (that is, to cells of category \( \beta \)).
For example, for
parameters of the cart-pole problem in the normal range (most
importantly, the update time interval and the magnitude of the
applied force) it is generally impossible for the system to
recover from a state with large values for which \( \theta = PM \) and
\( \dot{\theta} = PL \). Such fuzzy cells are essentially irrelevant.\(^8\) In an

\(^6\) It is important to note that parents are selected randomly “with
replacement” so that the same pair of parents are frequently picked a
number of times. The randomness of the crossover process then almost
always ensures that different pairs of children are produced in each
reproductive instance of the pair of parents.

\(^7\) If all gene strings in a generation had the same code in one particular position
then without the possibility of mutation it would not be possible for the code in
that position to change in all succeeding generations. If a code in one
particular position is missing then without mutation the missing code could
not enter the gene string at that position.

\(^8\) For problems with a large number of dimensions it is important where possible
to limit the number of relevant fuzzy cells in order to keep the computational
complexity within reasonable bounds. Then we would consider a sparse coding
of fuzzy cells rather than the dense coding we consider here.
GENETIC ALGORITHM FOR FINE-TUNING FUZZY RULES FOR THE CART-POLE BALANCING SYSTEM

attempt to avoid the distraction of having useless mutations the possibility of mutation is limited to those cells, called candidate cells or genes, that fire in any of the simulations during fitness evaluation of the parent generation. The sites of genes that are allowed to be mutated is thereby limited. The sites in the gene string of the genes among candidate genes and the mutated value among (-, n, o, p, +) are selected uniformly at random. In another variation of common practice, we ensure that a few of the fittest gene strings in a generation are propagated unchanged to the next generation. In this way we are assured that, at worst, the maximum fitness score in a generation does not decrease as the generations progress. We determine the fitness of each gene string by simulations of problems 1-3 running at 50, 30, and 20 Hertz for up to 2000 time steps in bang bang and continuous modes. That is, a single fitness score is calculated by summing the number of steps to failure for the eighteen problems defined above.

The algorithmic steps for our genetic algorithm are as follows:

Procedure Genetic_algorithm
Step 0: /* initialization */
read base_gene_string into new_generation;
read n = number of gene strings per generation;
read probabilities of crossover pc, pM, pM; /* eg 0.95 */ and
read specifications of simulation problems;
repeat engineer_mutation on base_gene_string
until new_generation is filled;
Step 1: /* fitness evaluation */
evaluate total fitness score of each gene string by
simulation of specified problems;
copy new_generation into generation sorted in
order of fitness;
tabulate array proportional_fitness_scores = fitness score of gene string / total fitness score for
generation;
Step 2: /* make decision to exit */
if a gene string has a perfect score then output
such gene string, signal success, and exit;
if user signals termination then output fittest gene
string, signal failure, and exit;
continue to step 3;
Step 3: /* generate next generation */
derive_next_generation (generation, new_generation);
return to step 1;

Procedure derive_next_generation (generation, new_generation)
/* procedure generates a new generation of gene strings in
new_generation from generation */
Step 1: copy a small number of the fittest gene strings from
generation into new_generation;
Step 2: select crossover with probability pc, otherwise
select mutation;
Step 3: if crossover then engineer_crossover to produce
2 new gene strings, else engineer_mutation to
produce 1 new gene string;
Step 4: repeat steps 2 and 3 until new_generation con­
tains n or n-1 gene strings;
Step 5: if new_generation contains n-1 gene strings then
engineer_mutation to produce 1 new gene string;

Procedure engineer_crossover
/* procedure returns 2 new gene strings for new_generation
formed by swapping prefixes of 2 selected gene strings in
generation */
Step 1: select 2 parent strings from generation at random
with probability in proportion to the
proportional_fitness_scores of the gene strings therein;
Step 2: select a crossover site randomly and uniformly
along the length of a gene string; /* eg between 1 and
38 */
Step 3: return 2 new gene strings formed by copying
swapped prefixes of gene strings selected in step
1 up to crossing point and copying (unswapped)
suffixes past crossing point;

Procedure engineer_mutation
/* procedure returns a single new gene string for
new_generation formed by random mutation(s) in a selected
gene string from generation */
Step 1: select a single parent string from generation at
random with probability in proportion to the
proportional_fitness_scores of its gene strings;
Step 2: select number of mutation sites randomly in
selected gene string based on probability pM;
Step 3: for each mutation select mutation site in parent
gene string uniformly and randomly among can­
didate cells or genes, /* eg 0.01 */
Step 4: for each mutation site in gene string choose
another code from set {-, n, o, p, +} uniformly and
randomly; /* eg if o at site select new value from
{-, n, p, +} */
Step 5: return mutated gene string;

6 GENETIC ALGORITHM FOR FUZZY MEMBERSHIP FUNCTIONS

Triangular membership functions are employed throughout. However, exterior triangles for each variable are degenerate being considered to stretch to infinity. The vertices of each triangle are coded into a “gene string” containing real numbers with the position in the gene string of the three vertices determining the membership function of a linguistic category of a fuzzy variable. Taking symmetry into account each of the variables $\theta$, $x$, $\dot{x}$, and the force $F$ requires three independent triangles while $\theta$ requires four. For interior triangles all three vertices are coded, while triangles for ZE categories are isosceles centred on 0 and occupy one position in the gene
string. For degenerate exterior triangles only two positions in the
gene string are needed. Each gene string therefore contains
33 real numbers. Crossover is only permitted between posi­
tions in the gene string where data for one triangle ends and
another begins. Otherwise, the engineering of crossover is
similar to the genetic algorithm for the fuzzy rules. The
selection of the number of mutations, the position in the gene
string of each mutation, and the sign of the mutation are
selected uniformly and at random. The amount of each muta­
tion is continuously variable and is determined by random
processes. Fitness assessment is identical with that for the
genetic algorithm applied to the rules.

One of the reasons for the success of fuzzy controllers
lies in the degree of latitude with which membership
functions can be specified provided that adjacent member­
ship functions overlap sufficiently yet not excessively,
and that the categories adequately cover the range of the
variables. After a new gene string is formed it needs to be
checked for acceptability in light of these restrictions.
Generally, a membership function for a linguistic category
is acceptable if the degree of overlap with adjacent catego­
ries lies in the range 0.2 to 0.3 and if each triangle does not
extend beyond the apex of adjacent triangles. Triangles
represented in gene strings are modified as necessary to
attempt to conform to these criteria. If it is not possible to
modify a gene string to become entirely acceptable then a
degree of overlap outside the desired range is accepted.

7 RESULTS
Fuzzy controllers capable of operating down to 20 Hz were
sought using the empirical fuzzy rules (see Figure 2) and
membership functions (see solid lines in Figure 1) as a starting
point. The genetic algorithm for the fuzzy rules was applied,
for a population of \( n = 20 \) gene strings per generation. Consider
one particular position in a population of gene strings. The
probability that a particular one of the 5 possible codes is
missing from all of \( n \) gene strings is \( (4/5)^n \). For \( n \) substantially
greater than 5, the probability that some code or codes are
missing from a particular position in the entire population is
approximately \( 5 \times (4/5)^n \). For a population of 20 gene strings
this equals 0.0576. For our gene strings of length 438 we can
expect a total of about 25 positions in the gene strings of an
entire generation where some code does not appear. To
counteract this we need to have a significant chance for
mutation to enable the appearance of absent codes. We started
with a probability of \( p_m = 0.9 \) for reproduction by crossover and
hence a probability of 0.1 of reproduction with a single parent.
For this we adopted for the probability of a mutation per gene
position of \( p_m = 0.1 \). Thus for a population of 20 we can
expect about a total of 90 mutations per generation.\(^9\)

It should be mentioned that at one stage the computa­
tions were interrupted in order to apply (off-line) the
genetic algorithm to fine-tune the empirical membership
functions. Those results, where they differ appreciably
from the empirical membership functions, are shown as the
broken lines in Figure 1. Henceforth the revised mem­
bership functions replace the empirical ones in all our
calculations. The genetic algorithm for the fuzzy rules was
then continued to obtain a gene string that balanced the
cart-pole system for all 18 simulation problems. This
required the evolution of 25 generations taking a few
hours of computer time on a 80486 PC. The revised rules
are shown in Figure 4. The differences between the em­
pirical rules of Figure 2 and the genetically derived rules
are substantial with about 14 percent being different. We
have shown in Figure 5 results of simulations in the form
of screen dumps with these genetically engineered rules
and membership functions for a selected problem. The
empty positions in the array of rules correspond to rules
that fire to some extent during the minute of simulation
time. Much the same subset of rules fire for all of our
problems and active rules amount to about 30 to 40 percent
of the total.

\[
\begin{array}{cccccc}
\hat{x} & \text{NM} & \text{NS} & \text{ZE} & \text{PS} & \text{PM} \\
---n+++ & ---n++n & ---nnn & ---n+ & +n & +n \\
+++++++ & +++++n & +n++ & ++n & ++n & +n \\
\text{NM} & ++++++ & ++++++ & ++++++ & ++++++ & ++++++ \\
+++++++ & ++++++ & ++++++ & +n & ++n & +n \\
+++++++ & ++++++ & ++++++ & +n & ++n & +n \\
\text{NS} & ++++++ & ++++++ & ++++++ & ++++++ & ++++++ \\
+++++++ & ++++++ & ++++++ & +n & ++n & +n \\
\text{ZE} & ++++++ & ++++++ & ++++++ & ++++++ & ++++++ \\
+++++++ & ++++++ & ++++++ & +n & ++n & +n \\
\text{PM} & ++++++ & ++++++ & ++++++ & ++++++ & ++++++ \\
+++++++ & ++++++ & ++++++ & +n & ++n & +n \\
\end{array}
\]

Figure 4: Rules obtained from the genetic algorithm applied to the
empirical set of rules of Figure 2.
GENETIC ALGORITHM FOR FINE-TUNING FUZZY RULES FOR THE CART-POLE BALANCING SYSTEM

Starting from the same initial set of empirical rules we increased the probability of crossover to \( p_c = 0.99 \) and decreased the mutation probability per gene to \( p_m = 0.01 \) while still maintaining a population of 20 strings per generation. We obtained a controller capable of comparable performance after only 18 generations. In a third set of calculations the genetic algorithm for the rules was applied with a starting population of 20 gene strings generated uniformly and randomly. Because no expert knowledge was used in this starting set we employed a large probability of mutation (0.1) in these calculations. A successful controller was found for simulations on the set of 18 problems after many hundreds of generations. This required about 40 hours of computer time. The set of rules obtained in this way differ from those of Figure 4 in about 75 percent of entries though many of these simply reflect the random starting categories of category A cells. Even greater computational effort was required to find a controller with all the rule consequents initially assigned the category ZE. The results show that the genetic algorithm search process for the fuzzy rules is much more efficient when an expert is used to specify an initial set of knowledgeable rules rather than an uninformed set.

With the evaluation process for gene string fitness discussed above there has been no assessment of quality of balance and hence no ability to influence it. The fitness evaluation process was modified to include a penalty component for simulations which neither fail nor become well-balanced. Specifically, we subtract a weighted sum of the root mean square (RMS) values of the four state variables over the latter half of a simulation from the fitness score thereby rewarding controllers that produce balance of good quality. With the rules of Figure 4 as a starting point, a new controller was easily found using the genetic algorithm which changed 18 percent of the rules. We have presented in Table 3 statistics to reflect the quality of balance obtained over 1000 secs of simulation time for selected problems using the empirical rules of Figure 2, the genetically engineered rules of Figure 4, and the new rules incorporating quality control. It was found that the introduction of this simple measure of quality control usually improves the quality of balance.

Because of the inbuilt antisymmetry of the fuzzy rules, any balance naturally takes place with the cart in the vicinity of the centre of the track. Balance around other places on the track can be easily achieved by a minor modification. A short time after the control experiment has started and initial strategies have brought about a measure of recovery from the initial conditions, typically a few seconds of simulation time, a simple bias is gradually applied to the value of \( x \) as time proceeds until the total bias equals the distance between the desired position of balance and the centre of the track. The fuzzy controller is then fooled into believing the required balancing position is the centre of the track. Provided the bias is applied gradually the fuzzy controller has no difficulty in coping.

### Table 3: RMS values of \( \theta, \hat{\theta}, x, \dot{x} \) for 1000 seconds simulation time.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Mode</th>
<th>Mode</th>
<th>Rule set</th>
<th>RMS values</th>
<th>( \theta )</th>
<th>( \dot{\theta} )</th>
<th>( x )</th>
<th>( \dot{x} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>50</td>
<td>EMP</td>
<td>0.31  11.55 0.19 0.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>50</td>
<td>EMP</td>
<td>0.43  3.19  0.19 0.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>50</td>
<td>GA</td>
<td>3.24  20.89 1.72 0.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>50</td>
<td>GA</td>
<td>6.05  21.24 0.64 0.77</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>50</td>
<td>GAQ</td>
<td>0.82  18.73 0.23 0.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>20</td>
<td>GA</td>
<td>1.61  29.66 0.31 0.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30/30/2/0</td>
<td>bb</td>
<td>20</td>
<td>GAQ</td>
<td>1.23  28.64 0.12 0.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/0/1.25/-1.5</td>
<td>bb</td>
<td>50</td>
<td>GA</td>
<td>1.69  15.35 1.61 0.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/0/1.25/-1.5</td>
<td>bb</td>
<td>50</td>
<td>GAQ</td>
<td>1.02  17.76 0.19 0.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/0/1.25/-1.5</td>
<td>co</td>
<td>50</td>
<td>GA</td>
<td>0.56  5.60 0.12 0.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/0/0/0</td>
<td>bb</td>
<td>50</td>
<td>GA</td>
<td>0.52  13.79 1.85 0.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/0/0/0</td>
<td>bb</td>
<td>50</td>
<td>GAQ</td>
<td>1.02  18.30 0.31 0.23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EMP empirical rules and ranges — Figures 2 and 1.
GA genetically engineered rules and ranges — Figures 4 and 1.
GAQ genetically engineered set of rules with quality.
bb bang bang simulation.
co continuous force simulation.
30/30/2/0 Problem 3 hardest initial configuration.
0/0/1.25/-1.5 Problem 2 medium hard initial configuration.
0/0/0/0 Basic initial configuration of static system.
Last 4 columns indicate degree of stability.
* Pole hit barrier after 206 secs.
8 CONCLUSIONS
The present investigation shows that it is practical and relatively straightforward to design a fuzzy controller for the cart-pole system provided some expert knowledge is available at the outset to produce a reasonable working set of fuzzy rules. The search for rules is not nearly as efficient if no expert knowledge is available for we found the computation time needed to produce a good controller was then excessive. The application of a genetic algorithm technique also improves the associated membership functions.

The results in Table 3 indicate that the best controllers obtained, as measured by the smallness of the RMS values of $\theta$ and $x$, are of good quality in that the pole is more or less well-balanced. The introduction of a criterion for quality in the fitness assessment algorithm tends to improve the controller. It was found that while the locations of the triangles specifying the membership functions are important only a single fine-tuning of the empirical membership functions was necessary using an off-line genetic type of algorithm.

It was shown that the one set of controlling fuzzy rules and membership functions can be used for a wide range of starting configurations for controllers operating down to 20 Hz in both bang bang and continuous mode. Moreover, it was found to be a trivial matter to allow for balance at a specified position on the track by gradually altering the cart’s perception of its position once the system has recovered from its starting configuration. It is expected that the methodology is applicable to other controllers.

Geva and Sitte (1993a) and (1993b) point out that meaningful comparisons of results for the cart-pole controllers developed by various workers are difficult. Accordingly, they have proposed a benchmark of tests; our results have been presented bearing in mind their suggested tests. Their own, investigations and re-calculations of the cart-pole system for each set of membership functions are important only a single fine-tuning of the empirical membership functions was necessary using an off-line genetic type of algorithm.

It was shown that the one set of controlling fuzzy rules and membership functions can be used for a wide range of starting configurations for controllers operating down to 20 Hz in both bang bang and continuous mode. Moreover, it was found to be a trivial matter to allow for balance at a specified position on the track by gradually altering the cart’s perception of its position once the system has recovered from its starting configuration. It is expected that the methodology is applicable to other controllers.

Geva and Sitte (1993a) and (1993b) point out that meaningful comparisons of results for the cart-pole controllers developed by various workers are difficult. Accordingly, they have proposed a benchmark of tests; our results have been presented bearing in mind their suggested tests. Their own, investigations and calculations of the cart-pole system for the BOXES scheme of Michie and Chambers (1968) and the adaptive heuristic critic scheme of Barto et al (1983) demonstrate that it is very difficult to design a controller providing a well-balanced cart-pole system for wide ranges of initial configurations and simulation frequencies. Geva and Sitte (1993a) and (1993b) point out that these and other learning systems do not provide as good a solution as the simple linear controller of equation (1) with its four weights easily found by simple unguided random search. This fact has brought into question the wisdom of using the cart-pole system as a test bed for controllers. Ironically, they also have shown that the weights for linear controllers themselves are difficult to find by learning processes. Nevertheless, it does not follow that fuzzy controllers for the cart-pole system are inherently easier to design than those for other comparable systems. The cart-pole problem requires a large number of fuzzy rules together with five sets of membership functions. We believe that the methods applied in the present investigation are generally applicable to other fuzzy controllers.

ACKNOWLEDGEMENTS
The author wishes to acknowledge the assistance of Mr Jeremy Zeller in some of the computer programming and in helping obtain the working set of fuzzy rules used as a starting point.

REFERENCES

BIOPGRAPHICAL NOTES
Gerard Finn is a Senior Lecturer at the Queensland University of Technology, Brisbane. He completed a PhD at the University of Queensland. Current research interests include fuzzy logic and neural networks.
The Power of List Ranking on a Reconfigurable Array of Processors with Wider Bus Networks*

Tzong-Wann Kao¹
Shi-Jinn Horng
Department of Electrical Engineering,
National Taiwan Institute of Technology,
43, Section 4, Kee-Lung Road, Taipei, Taiwan, R.O.C.

This paper makes an efficient improvement of processor complexity for computing list ranking and some related problems on a reconfigurable array of processors with wider bus networks by increasing the bus width between processors. Usually, the bus width of the system bus requires log N-bit for a parallel processing system with N processors; instead of using log N-bit, the bus width of the system bus of a reconfigurable array of processors with wider bus networks is assumed to be O(N^1/2)-bit, where ζ is a constant and ζ ≥ 1. Based on such an architecture and a base-n (n = N^1/2) number system technique, two constant time basic operations are first introduced for computing the prefix modular n and prefix division n of an N-bit binary sequence using N processors. Then, many fundamental algorithms can be solved in constant time on this newly created machine using N^(1+1/ζ) processors. These algorithms include the prefix sum of N integers problem, the unweighted (or weighted) list ranking problem, the Euler tour related problems and tree recursion related problems, respectively. Note that the last two categories of problems can be reduced to the list ranking problem. If this problem can be solved efficiently, then both of the others can also be solved efficiently. Another contribution of this paper is that the execution time of the proposed algorithms is tunable by the bus width of the system bus installed. That is, the wider the system bus installed, the faster the algorithm obtained.

Key Words and Phrases: list ranking, prefix sum, Euler tour, tree recursions, reconfigurable bus system, reconfigurable array of processors.

CR Categories: C.1.2, F.2.2, G.2.2.

1 INTRODUCTION

Recently, parallel computation models based on a reconfigurable bus system have received much attention due to their great computation and communication powers. A reconfigurable array of processors with wider bus networks (abbreviated to RAPWBN) is defined to be an array of processors connected to a set of reconfigurable bus systems whose configuration can be dynamically changed by setting up local switches for each processor to meet the data flow of the proposed algorithms. Many reconfigurable parallel processing systems similar to the RAPWBN have been created recently. These include the bus automaton as proposed by Rothstein (1976; 1988), the reconfigurable mesh as proposed by Miller et al (1988a), the polymorphic-torus network as proposed by Li and Maresca (1989a), the processor arrays with reconfigurable bus system (abbreviated to PARBS) as proposed by Wang and Chen (1990), the reconfigurable network as proposed by Ben-Asher et al (1991), and the reconfigurable array of processors (abbreviated to RAP) as proposed by Kao et al (1993a). All these models are functionally equivalent to the RAPWBN.

The RAPWBN is an interesting computation model announced recently. All researchers know that the PRAM (parallel random access machine) models such as the EREW (exclusive read exclusive write) model, the CREW (concurrent read exclusive write) model, and the CRCW (concurrent read concurrent write) model are too idealistic to be implemented by current technology. Algorithms developed in the PRAM models are very efficient, but only for theoretical interest. Usually, an algorithm developed in a more practical parallel processing system such as mesh-connected computers or tree machines will increase its time complexity due to global communications. To overcome the drawbacks caused by long distance communications, extra communications links are proposed in addition to the existing parallel system. Moreover, there are many algorithms that require their supported interconnection scheme to change during the execution time. Such an algorithm can run very well in any PRAM model but there is an extra cost in simulating a dynamic interconnection pattern for a fixed parallel architecture during execution time. The configuration of the RAPWBN is reconfigurable at run time and the system communications...
diameter can be reduced to 1. That is, the architecture of the RAPWBN can be dynamically changed during execution time by establishing local connection between system busses and processors. Such a characteristic deals not only with the dynamic interconnection pattern but also the global communications problems, simultaneously. In particular, unlike PRAM models, it is more realistic and can be implemented by VLSI technology, as shown by Li and Maresca (1989b) and Shu and Nash (1988).

One of the main drawbacks of the previously created reconfigurable parallel processing systems (Rothstein, 1976; 1988; Miller et al., 1988a; Li and Maresca, 1989a; Wang and Chen, 1990; Ben-Asher et al., 1991; Kao et al., 1993a) is that many processors are used to establish the system bus for a computation. Instead of using processors to establish a system bus configuration, we use the bus networks. Such a strategy can significantly reduce the number of processors used to solve a problem. Also, Li and Maresca (1989b) implemented a VLSI chip, called YUPPIE (Yorktown Ultra Parallel Polymorphic Image Engine), to evaluate the cost of the polymorphic-torus interconnection in terms of VLSI area. It is shown that by adding 20% silicon area over each processor, connection autonomy can be achieved. Here, connection autonomy means the ability of each processor to control the local switch at instruction level. This implies that it would be more efficient to save silicon area by increasing the bus network capacity rather than the processor complexity. This reason gives us considerable motivation to use a wider bus network to improve not only the performance of the reconfigurable parallel system but also to reduce its total VLSI silicon area.

A minor difference between the RAPWBN and other existing reconfigurable parallel processing systems (Rothstein, 1976; 1988; Miller et al., 1988a; Li and Maresca, 1989a; Wang and Chen, 1990; Ben-Asher et al., 1991; Kao et al., 1993a) is that the data bus between processors is assumed to be an M-row and N-column bus network. Also, the bus width of each bus network is assumed to be \( O(N^{1+\xi}) \)-bit, where \( N \) is the number of processors, \( \xi \) is a constant and \( \xi \geq 1 \). For simplicity, assume \( n = N^{1/\xi} \) and \( n \) is an integer. Indeed, many researchers have assumed that the bus capacity of their reconfigurable parallel systems was \( O(\log N) \)-bit, as there were at least \( \log N \) bit to represent the index of a processor for a parallel processing system with \( N \) processors (Rothstein, 1976; 1988; Miller et al., 1988a; Li and Maresca, 1989a; Wang and Chen, 1990; Ben-Asher et al., 1991; Kao et al., 1993a).

The algorithms that we present are used to illustrate the advantages of the RAPWBN over other proposed reconfigurable parallel processing systems. Based on this newly created machine and a based-n number system technique, we first introduce two basic operations for computing the prefix modular \( n \) and prefix division \( n \) of an \( N \)-bit binary sequence using \( N \) processors. Then, a lot of fundamental algorithms can be solved in constant time on this machine using \( N^{1+\xi} \) processors with \( N \times N^{1+\xi} \) bus networks. These algorithms include the prefix sum of \( N \) integers problem, the unweighted (or weighted) list ranking problem, the Euler tour related problems such as rooting a tree, preorder (or postorder) numbers of a tree, etc., and the tree recursion related problems such as computing the number of descendants of each vertex, computing the depth of each vertex and so on. Formerly, these problems were solved using \( N^2 \) processors. We reduce the processor complexity from \( N^2 \) to \( N^{1+\xi} \). Another contribution of this paper is that the execution time of the proposed algorithms is tunable by the bus width of the system bus installed.

The rest of this paper is organized as follows. We first describe the reconfigurable array of processors with wider bus networks upon which our algorithms are based in Section 2. Section 3 introduces two basic operations for computing the prefix modular \( n \) and prefix division \( n \) of an \( N \)-bit binary sequence. Section 4 deals with several \( O(1) \) algorithms which include the prefix sum of \( N \) integers problem, the unweighted (or weighted) list ranking problem, the Euler tour related problems and the tree recursion related problems, respectively. Finally, some concluding remarks are included in the last section.

### 2. The Computation Model and Basic Notations

A linear RAPWBN of size \( N \) contains \( N \) processors arranged in an \( M \)-row and \( N \)-column bus network. That is, the bus system can be thought of as logically arranged in an \( M \)-row and \( N \)-column network array. Each processor is identified by a unique index \( i \), \( 0 \leq i \leq N - 1 \). The \( M \)-row and \( N \)-column bus networks have \( 2MN \) ports and each port has a bus connection switches denoted by \(-S_i,j \), \(-S_i,j \), \( 0 \leq i \leq M - 1 \), \( 0 \leq j \leq N - 1 \) and \(-S_i,j \), \( +S_i,j \), \( 0 \leq k \leq n - 1 \), respectively. The \( i \)-th row bus network connects the \( j \)-column port switch \(+S_i,j \) to \((j+1)\)-column port switch \(-S_i,j+k \), \( 0 \leq i \leq M - 1 \), \( 0 \leq j \leq N - 1 \). Each processor \( P_i \) also has an \( n \)-bit column bus with \( M \) ports and each port has \( n \)-bit connection switches denoted as \#S_i,j \( \times (k) \), \( 0 \leq i \leq M - 1 \), \( 0 \leq k \leq n - 1 \), respectively. The \( n \)-bit column bus of a processor can connect to the row bus network by setting the port connection switches \#S_i,j \( (k) \), \(-S_i,j+k \), \( 0 \leq i \leq M - 1 \), \( 0 \leq j \leq N - 1 \), \( 0 \leq k \leq n - 1 \), to be connected together. By setting \#S_i,j \( (k) \), \(-S_i,j+k \), \( 0 \leq i \leq M - 1 \), \( 0 \leq j \leq N - 1 \), \( 0 \leq k \leq n - 1 \), to be connected together, we show in Figure 1 an example of a linear RAPWBN of size \( 4 \) with \( 4 \)-row and \( 4 \)-column bus networks and a bus width for each bus network of \( 2 \)-bit.

For a unit of time, assume each processor can perform one of the following operations: execute arithmetic and logic operations, access a local memory word, connect or disconnect a bus port, set or reset all connection switches on the same column bus or communicate with others by broadcasting data on a bus. It allows multiple processors to broadcast data on the different buses or to broadcast the same data on the same bus simultaneously at a time unit, if there is no collision.
The power of list ranking on a reconfigurable array of processors with wider bus networks

Let \( var(k) \) denote the local variable \( var \) (memory or register) of a processor with index \( k \). For example, \( sum(1) \) is a local variable \( sum \) of processor \( P_1 \).

A RAPWBN is operated in a single instruction stream, multiple data streams (SIMD) model. The bus bandwidth is not unlimited between processors. We assume the bus bandwidth is bounded by \( n \)-bit, where \( n \) is an integer. A constant time can be achieved for transferring an \( n \)-bit of data between processors under such an assumption. The I/O loading (up load or down load) time is fully dependent on how complex the I/O interface between processors and peripherals will be. For example, Ben-Asher et al (1991) used an initialising network to deliver the input to the switches of a 2-D reconfigurable mesh. Therefore, the complexity of an algorithm is assumed to be the sum of the maximal computation time among all processors and the communication time among all processors. This assumption was also used by Ben-Asher et al (1991); Li and Maresca (1989a; 1989b); Miller et al (1988a; 1988b); Olariu et al (1991; 1992); Park et al (1993) and Wang et al (1990; 1991).

3 BASIC OPERATIONS AND ALGORITHMS

Some data operations will be described in this section. These data operations are used for deriving several efficient basic algorithms in the following sections. Assume \( \text{div} \) and \( \text{mod} \) are the integer division and modular operators. The integer remainder sequence of the prefix modular \( n \) computation and the binary quotient sequence of the prefix division \( n \) computation of a binary sequence \( b_0, b_1, \ldots, b_{N-1} \) are the two problems to be considered first; that is, computing \( r_j = (\Sigma_{i=0}^{j} b_i) \mod n \) and \( q_j = (b_j + r_{j-1}) \div n \), for \( 0 \leq j \leq N - 1 \), respectively. Based on the configurational computation approach as shown by Wang et al (1991); Park et al (1993) proposed an \( O(1) \) time algorithm for the prefix modular \( n \) computation of an \( N \)-bit binary sequence on a reconfigurable mesh using \( (n + 1) \times 2N \) processors. The main idea for this problem as proposed by Park et al (1993) consists of two steps. First, establish a configurational prefix modular \( n \) computation bus using \( (n + 1) \times 2N \) processors. Then, compute the prefix modular \( n \) using the computation bus as established previously.

However, the reconfigurable mesh model proposed by Park et al (1993) is quite different from ours. Instead of using processors to construct the configurational computation bus, we use the bus networks to establish it. Such an approach leads to a reduction of processor complexity to \( N \) but still keeps the same time complexity as that proposed by Park et al (1993). As demonstrated by Li and Maresca (1989b), this approach can save quite a lot of total VLSI silicon area. We solve these two problems as proposed by the following lemmas. Assume \( n = 3 \), an example for the prefix modular \( n \) computation and the prefix division \( n \) computation of a binary sequence \((0, 1, 1, 0, 1, 0, 0, 1, 1)\) is shown in Figure 3.
Let $q_h$ be the integer division, $q_h = (b_k + r_{k-1}) \div n$ can be also represented as

$$q_h \times n + r_k = b_k + r_{k-1}$$

for $0 \leq k \leq N - 1$, where $r_{-1} = 0$. Then,

$$\sum_{k=0}^{j} (q_h \times n + r_k) = \sum_{k=0}^{j} (b_k + r_{k-1})$$

for $0 \leq j \leq N - 1$.

Hence, $\sum_{k=0}^{j} q_h = (\sum_{k=0}^{j} b_k - r_j) \div n$, for $0 \leq j \leq N - 1$.

For Boolean operation, there was a result derived by Miller et al. (1988a; 1988b).

**Lemma 3**

Miller et al. (1988a; 1988b): Given a linear reconfigurable mesh of size $N$, in which each processor stores a piece of Boolean data, the logical OR of these data can be computed in $O(1)$ time.

### 4 APPLICATIONS

In this section we discuss several applications that are based upon the basic operations proposed in the previous section.

#### 4.1 Prefix Sum of N Integers

Let $psum_j$ be the prefix sum of $N$ (log $N$)-bit integers and it is defined as

$$psum_j = \sum_{i=0}^{j} A_i, \quad (1)$$

where $0 \leq A_i < N$ and $0 \leq j < N$. Based on the prefix sum of a binary sequence, Olariu et al. (1992) proposed an $O(1)$ time algorithm for this problem on reconfigurable meshes using $N \times N$ processors. In this paper, we use another approach to solve this problem on a linear RAPWBN using $N^{1+1/\xi}$ processors with $N^{1/\xi}$-bit bus width, where $\xi$ is a constant and $\xi \geq 1$. Since $A_i < N$, $0 \leq i < N$, and $psum_j$, $0 \leq j < N$, is at most $N(N-1)$. Using the base-$n$ number system technique, $A_i$ and $psum$, can be represented as follows.

$$A_i = \sum_{k=0}^{r_{i-1}} a_{i,k} \cdot n^k, \quad (2)$$

$$psum_j = \sum_{i=0}^{r_{j-1}} e_{j,i} \cdot n^i, \quad (3)$$

where $0 \leq A_i < N$ and $0 \leq j < N$. Based on the prefix sum of a binary sequence, Olariu et al. (1992) proposed an $O(1)$ time algorithm for this problem on reconfigurable meshes using $N \times N$ processors. In this paper, we use another approach to solve this problem on a linear RAPWBN using $N^{1+1/\xi}$ processors with $N^{1/\xi}$-bit bus width, where $\xi$ is a constant and $\xi \geq 1$. Since $A_i < N$, $0 \leq i < N$, and $psum_j$, $0 \leq j < N$, is at most $N(N-1)$. Using the base-$n$ number system technique, $A_i$ and $psum$, can be represented as follows.
where \( T = \lfloor \log_n N \rfloor + 1, 0 \leq i < N, 0 \leq k \leq T - 1, T' = \lfloor \log_n N \rfloor \), \( 0 \leq j < N, 0 \leq i \leq T - 1, 0 \leq a_{i,k} \leq n - 1 \). As \( psum_j = \sum_{i=0}^{T-1} \sum_{k=0}^{i} d_{i,k} n^k \), let \( d_{i,k} \) be the prefix sum of \( N \) coefficients \( a_{i,k}, 0 \leq i < N \), which is defined as follows.

\[
d_{i,k} = \sum_{i=0}^{k} a_{i,k}, \tag{4}
\]

where \( 0 \leq j < N, 0 \leq k \leq T - 1 \). Then \( psum_j \) can be also represented as

\[
psum_j = \sum_{i=0}^{T-1} d_{i,k} n^k, \tag{5}
\]

where \( 0 \leq d_{i,k} \leq (n-1)(j+1) \). Let \( C_{j,0} = 0 \) and \( d_{j,n} = 0 \), \( T \leq u \leq T' - 1 \). The relationship between Eq. (3) and Eq. (5) is described by Eqs. (6)-(9).

\[
S_{j,t} = C_{j,t} + d_{j,t}, \quad 0 \leq t \leq T' - 1, \tag{6}
\]

\[
C_{j,t} = S_{j,t-1} \mod n, \quad 1 \leq t \leq T' - 1, \tag{7}
\]

\[
e_{j,t} = psum_j \mod n, \quad 0 \leq t \leq T' - 1, \tag{8}
\]

where \( S_{j,t} \) is called the sum at the \( t \)th digit position and \( C_{j,t} = S_{j,t-1} \mod n \) is called the carry to the \( t \)th digit position. Hence, \( e_{j,t} = S_{j,t} \mod n \) is the coefficient of \( psum_j \) under the base-\( n \) system. Since the carry to the \( t \)th digit position of \( S_{j,t} \) is not greater than \( j \), we have \( C_{j,t} = j, 0 \leq j < N, 0 \leq t \leq T' - 1 \). Let \( q_{i,k} \) be the binary quotient sequence of the prefix division \( n \) computation on an integer sequence \( a_{i,k} \), (which is represented by an \( n \)-bit binary sequence). \( C_{j,t} \) can also be obtained by

\[
C_{j,t} = \sum_{i=0}^{t} q_{i,t}, \tag{9}
\]

Therefore, instead of computing Eq. (1), we first compute the coefficient \( a_{i,k} \) for each \( A_{i} \). Then each \( e_{j,t} \) can be computed by Eqs. (4), (6)-(9). Finally, each \( psum_j \) can be computed by Eq. (3). Assume the bus width is \( n \)-bit, where \( n = N^{\lceil \log_n N \rfloor} \) is an integer and \( \zeta \geq 1 \). Then the prefix sum of \( N \) \((\log N)\)-bit integers can be solved in constant time on an RAPWNB using \( N^{\lceil \log_n N \rfloor} \) processors and a row bus network with \( N^{\lceil \log_n N \rfloor} \)-bit bus width. For the sake of readability, let \( P_{i,r} \), \( 0 \leq u \leq n - 1, 0 \leq v \leq n - 1 \), denote the logical processor corresponding to the physical processor \( P_{i,r} \), \( n = N^{\lceil \log_n N \rfloor} \) and \( \zeta \geq 1 \). The prefix sum of \( N \) integers algorithm (PSNIA) is listed as follows. Assuming \( n = 3, A_{0} = 2, A_{1} = 1 \) and \( A_{2} = 2 \), a snapshot of procedure PSNIA for the prefix sum of 3 integers is shown in Figure 4.

**procedure PSNIA(\( A_{i} \), \( psum_{i} \));**

**Input:** An integer sequence \( A_{i} \) is stored in the local variable \( A(i) \) of processor \( P_{i,0} \), \( 0 \leq i \leq N - 1 \).

**Output:** The integer prefix sum sequence \( psum_{i} \) is stored in the local variable \( psum(i) \) of processor \( P_{i,0} \), \( 0 \leq i \leq N - 1 \).

**Step 0. begin**

**Step 1. for** \( 0 \leq i < N \) **do in parallel begin**

**Step 1.1 Processor** \( P_{i,0} \), **copies** \( A(i) \) **to processor** \( P_{i,j} \), \( 0 \leq i \leq N - 1, 0 \leq j \leq n - 1 \).

**Step 1.2 Processor** \( P_{i,j} \), **sets** \( q(i,0) = 0 \).

**end (for);**

**Step 2. for** \( 0 \leq k \leq T' - 1 \ **do begin**

**Step 2.1 Processor** \( P_{i,j} \), \( 0 \leq i \leq N - 1, 0 \leq j \leq n - 1 \), **computes** \( a(i,k) \) **from** \( A(i) \) **by using Eq. (2).**

**Step 2.2 Processor** \( P_{i,j} \), **sets** \( b(i,j) = 1 \) **if** \( j \neq a(i,k) \); **otherwise**.

**Step 2.3 Processor** \( P_{i,j} \), **sets** \( b(i,j) = 0 \) **for** \( i \leq N - 1, 0 \leq j \leq n - 1 \), **sets** \( q(i,j) = q(i,k) \).

**Step 3. (b) After Step 2.1 for** \( k = 0 \) **case.**

**Step 3.2 Processor** \( P_{i,j} \), **sets** \( q(i,0) = 0 \) **for** \( i \leq N - 1, 0 \leq j \leq n - 1 \), **sets** \( q(i,j) = 0 \) **for** \( i \leq N - 1, 0 \leq j \leq n - 1 \), **saves** \( psum(i) \).

**end (for);**

**Step 3.3 (a) Initialization.**

**Step 3.4 (cl) After Step 2.4 for** \( k = 0 \) **case.**

**Step 3.5 (c) After Step 2.4 for** \( k = 0 \) **case.**

**Step 3.6 (d) After Step 2.4 for** \( k = 0 \) **case.**

**Step 3.7 (e) After Step 2.1 for** \( k = 1 \) **case.**

**Step 3.8 (a) Initialization.**

**Step 3.9 (b) After Step 2.1 for** \( k = 1 \) **case.**

**Step 3.10 Processor** \( P_{i, j-1} \), **sets** \( q(i,0) = 0 \).

**end (for);**

**Step 3.11 Compute** \( e(i,k) \) **obtained from Step 2, processor** \( P_{i,j-1} \), **computes Eq. (3) to obtain the** \( psum(i) \), **for** \( 0 \leq i \leq N - 1 \).

**end (for);**

**Step 3.12 Using** \( e(i,k) \) **obtained from Step 2, processor** \( P_{i,j-1} \), **computes Eq. (3) to obtain the** \( psum(i) \), **for** \( 0 \leq i \leq N - 1 \).
Step 3.2 Processor \( P_i, \ 0 \leq i \leq \log A \), copies \( psum(i) \) to processor \( P_i, \ 0 \leq i \leq N \).

Step 4. end; {procedure PSNIA}

Theorem 1
The PSNIA procedure can be computed in \( O(T') \) time, where \( T' = \log_\log A(N-1) + 1 \), on a linear RAPWBN of size \( N \) with \( n \)-bit bus width for \( n \geq 2 \).

Proof:
The correctness of procedure PSNIA directly follows from Lemmas 1-2 and Eqs. (1)-(9). The time complexity is analysed as follows. By Eqs. (1) and (3), \( psum_i \) is at most \( N(N-1) \); it requires at most \( \log_\log A(N-1) + 1 \) digits to represent \( psum_i \) in a base-\( n \) number system. Thus, the number of iterations \( T' \) of Step 2 is \( \log_\log A(N-1) + 1 \). Each iteration of Step 2 takes \( O(1) \) time. Step 1 takes \( O(1) \) time. Finally, Step 3 computes Eq. (3) in \( O(T') \) time. Hence, the total time complexity is \( O(T') \).

By assumption, \( n \) is \( \sqrt{A} \). If \( \xi \) is a constant, then \( T' = \log_\log A(N-1) + 1 = 2\xi^N + 1 \) is also a constant. This leads to the following corollary.

Corollary 1
Given \( N \log (\log N) \)-bit integers, the prefix sum of these \( N \) integers can be computed in \( O(1) \) time on a linear RAPWBN of size \( N^{1+1/\xi} \) with \( \log A \)-bit bus width, where \( \xi \) is a constant and \( \xi \geq 1 \).

Note that Theorem 1 implies that the execution time of the PSNIA algorithm is tunable by the system bus width. The wider the system bus installed, the better the execution time of procedure PSNIA obtained. Even though the system bus width is reduced to \( O(\log N) \)-bit, it is still quite efficient. In the following, except when otherwise noted, the bus width of each bus network is assumed to be \( O(N^{1/\xi}) \)-bit, where \( \xi \) is a constant and \( \xi \geq 1 \).

In particular, if \( A_i \) is a binary number, then procedure PSNIA can compute a binary sequence of length \( N \) in \( O(1) \) time on a linear RAPWBN of size \( N \) with a row bus network.

Two similar results for the prefix sum of a binary sequence can be found in Kao et al. (to appear) and Lin (1992). This leads to the following corollary.

Corollary 2
Kao et al. (to appear) and Lin (1992): Given a binary sequence of length \( N \), the prefix sum of these \( N \) binary elements can be computed in \( O(1) \) time on a linear RAPWBN of size \( N \) with a row bus network.

4.2 The Unweighted List Ranking
Given a linked list \( a_0, a_1, ..., a_N \) of \( N \) elements with \( a_i \) following \( a_{i-1} \) in the list, the unweighted list ranking problem is to determine the rank (distance) from each element to the head of the list. The list ranking problem discussed here is the data dependent version. That is, the location of the elements are given but it is not known which element is which. Only the location of the first element is given along with a map from the \( i^{th} \) element to the \( (i+1)^{th} \) element. In practice, the mapping will be represented as a linked list, so that if element \( i \) is contained in the base-\( N \) number system separately or performing the computation in Step 2 of procedure PSNIA in \( T' = \log_\log A(N(N-1) + 1 \) iterations under the base-\( n \) number system. By Corollary 1, this takes \( O(\xi N) \) time on a linear RAPWBN of size \( N^{1+1/\xi} \). Since \( \xi \geq 1 \) is also a constant, this leads to the following corollary.

Corollary 3
Given a sequence of \( N \log N \)-bit integers in the range from 0 to \( N^\xi - 1 \), the prefix sum of these \( N \) integers can be computed in \( O(1) \) time on a linear RAPWBN of size \( N^{1+1/\xi} \) with a row bus network.

Note that the problem of computing the prefix sum of \( N \log (\log N) \)-bit signed integers bounded within \([-N, N] \) can be easily reduced to the problem of computing the prefix sum of \( N \log N \)-bit unsigned integers. Finally, consider the problem of finding the maximum of \( N \) integer numbers \( A_0, A_1, ..., A_{N-1} \). Recently, Kao and Horng (1994) proposed a constant time algorithm for this problem on an RAPWBN using \( N \) processors with a row bus network.
jth processor then element $i+1$ will be contained in the $next(j)$ processor, where $next(j)$ is the link of element $i$. If $next(j) = -1$ then element $i$ is the tail of the list and the ranks of element $i$ and the head are $N-1$ and 0, respectively.

The list ranking problem has been studied extensively by several researchers in different computation models (Cole and Vishkin, 1986; 1989; Han, 1987; Kruskal et al, 1985; 1986a; Olariu et al, 1991; Ryu and Jajodia, 1990; Subbaraman et al, 1993; Vishkin, 1984; Wyllie, 1979). For a reconfigurable array of processors, Olariu et al (1991) first proposed an $O(1)$ time algorithm for this problem on 2-D reconfigurable meshes using $(N^2(N+1)+2) \times 3N$ processors, for any constant $0 < \epsilon \leq 1$. Later, based on the wider bus architecture, Lin and Olariu (1993) extended the bus width of the system bus to $N^{1/6}$-bit ($\xi$; a constant and $\xi \geq 1$) and proposed an $O(1)$ time algorithm for this problem on the 2-D reconfigurable meshes using $N^2$ processors. Subbaraman et al (1993) proposed another $O(1)$ time algorithm for this problem using $N^{1+\epsilon}$ processors and $N^N$ bus networks.

The list ranking algorithms proposed by Lin and Olariu (1993) and Subbaraman et al (1993) are based on the same idea and consist of two major steps. First, embed the linked list in the proposed computation model. Then, compute the rank of each element in the linked list by the prefix sum algorithm as mentioned in Corollary 2. Based on the wider bus network approach, the above two list ranking algorithms proposed by Lin and Olariu (1993) and Subbaraman et al (1993) can be improved and run in the same time complexity while reducing the processor complexity to $N$. This is an efficient approach for improving the list ranking algorithm under an RAPWBN. The main idea of our algorithm is to reduce the unweighted list ranking problem to the binary sequence prefix sum problem. Given a linked list $a_0, a_1, ..., a_{N-1}$ of $N$ elements with $a_{i+1}$ following $a_i$, assume that $a_i$ and $a_{i+1}$ are in processors $P_i$ and $P_{next(i)}$, respectively. That is, the link of element $i$ is stored in the local variable $next(j)$ of processor $P_j$. Finally, the rank of element $a_i$ is stored in the local variable $rank(j)$ of processor $P_j$. The list ranking algorithm consists of three major steps. Let $a_0$ be stored in processor $P_0$. Assuming $next(0) = 2$, $next(1) = -1$, $next(2) = 3$ and $next(3) = 1$, a snapshot of the computing of the rank of each element of the linked list is shown in Figure 5.

**Step 1.** Processor $P_j$, $0 \leq j \leq N-1$ sets $b(j) = 1$, if $next(j) \neq -1$; $b(j) = 0$, otherwise.

**Step 2.** for $0 \leq i \leq \lceil \xi \rceil$ do begin

**Step 2.1** Processor $P_j$, $0 \leq j \leq N-1$ sets the port connections $\{\#S_{next(j)}, -S_j, j, +S_j, j\}$, and sets the port connections $\{\#S_{next(j)}, (k+1) \mod N^{\lfloor \xi \rfloor}, -S_{next(j)}, j, +S_{next(j)}, j, (k+1) \mod N^{\lfloor \xi \rfloor}\}$.

**Step 2.2** Compute the integer remainder $r(j, i)$ from port $-S_j, j$ (or $+S_j, j$) and the binary quotient $q(j, i)$ from port $-S_j, j$ (or $+S_j, j$).

**Step 2.3** Processor $P_j$, $0 \leq j \leq N-1$ sets $b(j) = q(j, i)$. end for;

**Step 3.** Processor $P_j$, $0 \leq j \leq N-1$, computes $rank(j) = \sum_{i=0}^{\lceil \xi \rceil} r(j, i)$. Note that the above algorithm takes constant time by Corollary 2. Therefore, our list ranking algorithm runs in constant time on an RAPWBN using $N$ processors with $N$-row and $N$-column bus networks. This leads to the following lemma.

**Lemma 5**

The unweighted list ranking algorithm can be executed in $O(1)$ time or a linear RAPWBN of size $N$ with $N$-row and $N$-column bus networks.

**Proof:**

The correctness of this algorithm follows from Lin and Olariu (1993) and Subbaraman et al (1993). That is because the rank of element $j, 0 \leq j < N$, is equal to its prefix sum value. The time complexity is $O(1)$ by Corollary 2.
Instead of computing the rank of each element of the list, we are interested in computing the prefix sum of the list. This problem is also denoted as the weighted list ranking problem. Assume each data element is represented by either an \(O(\log N)\)-bit integer or a real number.

The algorithm for computing the prefix sum of a linked list consists of four major steps. Step 1, compute the rank of each element of the linked list by setting each element with a weight 1 by Lemma 5. Then, each element is associated with a rank and each rank is associated with a corresponding row bus network. Step 2, rearrange each element of the linked list in an ascending order by using the corresponding row bus network. Step 3, compute the prefix sum of the \(N \cdot O(\log N)\)-bit newly created linked list using the prefix sum algorithm by Corollary 3 or Corollary 4. Finally, the prefix sum can be routed back to its original location by the reverse operation of Step 2. Therefore, we have the following theorem.

**Theorem 2**

The prefix sum of a linked list with \(N \cdot O(\log N)\)-bit integers or real numbers can be computed in \(O(1)\) time on an RAPWBN using \(N^{1+\epsilon}\) processors with \(N\)-row and \(N^{1+\epsilon}\)-column bus networks.

### 4.3 The Euler Tour and Related Problems

In this subsection we will discuss the Euler tour and some related problems.

**Lemma 6**

Kao et al. (1993b): Given a linear RAPWBN of size \(N\) with a row bus network, in which each processor \(P_i\) has a Boolean item \(c_i\), where \(c_i\) is 0 or 1 and \(0 \leq i \leq N - 1\), the linked list of these nonzero items (including a head and a tail) can be created in \(O(1)\) time.

From Lemma 6, it is easy to obtain the circular linked list of a binary sequence by setting \(\text{next}(\text{tail}) = \text{head}\).

**Lemma 7**

Wang and Chen (1990): Given a linear RAPWBN of size \(N\) with a row bus network, in which each processor \(P_i\) has a Boolean flag \(f_i\), \(0 \leq i \leq N - 1\), we can determine in \(O(1)\) time if there exist processors whose value of \(f_i\) is 1, and the processor with minimal (maximal) index, if more than one processor has \(f_i = 1\).

Let \(T = (V, E)\) be a tree which is represented by an adjacency list, and let \(T' = (V, E')\) be the directed graph obtained from \(T\) when each edge \((u, v) \in E\) is replaced by two arcs \((u, v)\) and \((v, u)\). Since the indegree of each vertex of \(T'\) is equal to its outdegree, \(T'\) is an Eulerian graph; that is, it has a directed circuit that traverses each arc exactly once. An Euler tour of a graph is a cycle that traverses each edge of the graph exactly twice. In general, we say that the Euler circuit of \(T\) refers to the Euler tour of \(T\). An Euler circuit of \(T' = (V, E')\) can be defined by specifying the successor function \(s\) mapping each arc \(e \in E'\) into the arc \(s(e) \in E'\) that follows \(e\) on the circuit.

Given a tree \(T = (V, E)\), and an ordering of the set of vertices adjacent to each vertex \(v\), an Euler circuit of the directed graph \(T' = (V, E')\) can be easily specified. The Euler tour algorithm is based on the method which was proposed by Jalka (1992) and Tarjan and Vishkin (1985).

First, a tree \(T\) consists of the adjacency lists of the vertices; for each vertex \(v \in V\), let \(L[v]\) be its adjacent neighbours and let them be given in a circular linked list. That is, \(L[v] = \langle u_0, u_1, \ldots, u_{d-1} \rangle\), is enumerated in some order, where \(d\) is the degree of \(v\). Next, each edge \((u, v)\) chooses the edge \((v, u)\) as the successor of \((u, v)\) to construct an Euler tour of \(T\) using the ordering implied by the adjacency list, where \(u_i\) is the \(i^{th}\) vertex of \(L[v]\), and \(0 \leq i \leq d - 1\).

Let \(l(<u, v>)\) be the successor location function which points to the processor that holds the successor of arc \((u, v)\). That is, if processor \(P_i\) holds an arc \((u, v)\) then the successor location function \(l(<u, v>) = j\). An Euler tour of \(T\) can be constructed in \(O(1)\) time on an RAPWBN using \((2N - 2)\) processors with \(N\)-row and \((2N - 2)\)-column bus networks. Assume the \(N - 1\) edges of \(T\) are loaded in processors \(P_j\), \(0 \leq j \leq N - 2\). The Euler tour algorithm consists of four major steps. Given a tree \(T\) as shown in Figure 6. (a), assume the four edges \((1, 0), (2, 1), (3, 1)\) and \((0, 4)\) are stored in processors \(P_0, P_1, P_2, P_3\), respectively. A snapshot of the creating of an Euler tour of a tree \(T\) is shown in Figure 6. Step 1, processor \(P_j\), \(0 \leq j \leq N - 2\), which holds edge \((u, v)\), creates a pair of oppositely directed edges \((u, v)\) and \((v, u)\). Step 2, processor \(P_j\), \(0 \leq j \leq N - 2\), which holds edge \((u, v)\), broadcasts the oppositely directed edge \((v, u)\) to processor \(P_{j+N-1}\) using the \(j^{th}\) row bus network. Step 3, obtain the adjacent circular linked list \(L[v]\) using the \(v^{th}\) row bus network, for each vertex \(v \in V\) by Lemma 6. That is, processor \(P_j\), \(0 \leq j \leq 2N - 2\), which holds edge \((u, v)\), uses the \(v^{th}\) row bus network to construct the circular linked list \(L[v]\) and the link is stored in the local variable next. Finally, processor \(P_j\), \(0 \leq j \leq 2N - 2\), obtains the successor function \(s(<u_j, v>) = \langle v, u_{(i+1)\mod N} \rangle\) and the successor location function \(l(<u_j, v>) = \langle v, u_i \rangle\). That is, processor \(P_j\), \(0 \leq j \leq N - 2\) (or \(P_j, N - 1 \leq j \leq 2N - 3\), which holds edge \((u_j, v)\) (or edge \((v, u_i, v)\)), uses the \(j^{th}\) row bus network to get the local variable next(<v, u_i>) (or next(<u_j, v>)) of processor \(P_{j+N-1}\) (or \(P_{j-N+1}\)). This leads to the following theorem.

**Theorem 3**

An Euler tour of a tree \(T = (V, E)\) with \(N\) vertices which is represented by a set of edges can be created in \(O(1)\) time on an RAPWBN using \((2N - 2)\) processors with \(N\)-row and \((2N - 2)\)-column bus networks.

**Proof:**

The correctness of this algorithm follows from Jalka (1992) and Tarjan and Vishkin (1985). The time complexity is analysed as follows. Steps 1, 2 and 4 take \(O(1)\) time, respectively. Step 3 takes \(O(1)\) time by Lemma 6. Hence, the time complexity is \(O(1)\).
Applying the list ranking algorithm and prefix sum of a list algorithm on an Euler tour of a tree $T$, several tree problems such as rooting a tree, computing the preorder and postorder numbers and finding the vertices order while visiting the Euler tour of $T$ can be solved in constant time on an RAPWBN using $O(N)$ processors with $O(N)$-row and $O(N)$-column bus networks. We start with the problem of rooting a tree $T$ at vertex $r$. That is, for each vertex $v \neq r$, we will determine the parent $p(v)$ of $v$ when $T$ is rooted at $r$. Let the adjacency list of vertex $r$ be given by $L[r] = [u_0, u_1, \ldots, u_{|E|}]$. The problem of rooting a tree $T$ can be solved by the following three steps. Step 1, break the Euler tour at $r$ by setting $s(\langle u_0, r \rangle) = -1$ (i.e. nil) to form an Euler path $EP$. Step 2, visit exactly once each arc of the Euler path $EP$ and find the vertices order while visiting the Euler tour of $T$. Step 3, for each arc $\langle u, v \rangle$, assume $i$ is the rank of $\langle u, v \rangle$ on the Euler path $EP$, then set $A(i) = u$. Finally, set $A(2N-2) = r$.

The detailed algorithms for the above discussed problems can be found in those proposed by Jajčá (1992).

Theorem 4
Given a tree $T = (V, E)$ with $N$ vertices and rooted at $r$, which is represented by a set of edges, the following problems can be solved in $O(1)$ time on an RAPWBN using $O(N)$ processors with $O(N)$-row and $O(N)$-column bus networks.

1. Rooting a tree.
2. Computing preorder and postorder numbers of each vertex.
3. Finding the vertices order while visiting the Euler tour of $T$.

4.4 Tree Recursions
Let $T = (V, E)$, rooted at $r$, $|E| = |V| - 1$, be a tree which is represented by a set of edges. Then the prefix sum on a tree is defined as

$$pst_u = \sum_{v \text{ child of } u} val_v, \quad (10)$$

where $val_v$ and $pst_u$ are two $O(\log N)$-bit integers or real numbers.

This problem had been studied for mesh connected computers by Atallah and Hambrusch (1985) and Stout (1985). Then Kruskal et al (1986b) presented an $O((N \log N)(P \log(NP)))$ prefix sum algorithm for a tree with $N$ vertices on an EREW machine with $P$ processors and $P < N$.

To solve the tree recursions problem on a linear RAPWBN, we need the following lemma.

Lemma 8
Jajčá (1992): Given a rooted tree $T = (V, E)$, let $A$ be the array of vertices order as visited by the Euler tour of $T$, and $l(v)$ and $r(v)$ be the indices of the leftmost and rightmost appearances of $v$ in $A$, for $v \in V$. Let $u$ and $v$ be two arbitrary distinct vertices of $T$. Then $u$ is an ancestor of $v$ if and only if $l(u) < l(v) < r(u)$.

Let $p(v)$ be denoted the parent vertex of vertex $v$. We also assume that the $N - 1$ edges of a tree are stored in the local variable $e(i)$ of processor $P_i$, $0 \leq i \leq N - 2$. The tree recursions algorithm consists of six major steps. Given a tree $T$ as shown in Figure 7. (a), assume $val_0 = 5, val_1 = 2, val_2 = 4, val_3 = 3$ and $val_4 = 1$, respectively. A snapshot of the computing of the tree recursions of a tree rooted at vertex 0 is shown in Figure 7. Step
1, create an Euler tour of \( T \). Step 2, compute the array \( A \) of vertices order as visited by the Euler tour of \( T \). Step 3, root the tree \( T \). That is, find the parent vertex of each vertex. Step 4, assign the weights \( w(<v, p(v)>)=\text{val}_v, w(<p(v), v>)=0 \). Step 5, compute the array \( B, B(1) \) to \( B(2N-2) \), by performing the parallel prefix sum on the weighted Euler tour of \( T \), initially, \( B(0)=0 \). Step 6, for each vertex \( u \), compute the prefix sum on a tree \( T \), where of \( \text{ pst}_u=\sum_{v} \text{child}_o(v) \text{val}_v \). We only need to show the details of the implementation of Step 6 on the proposed RAPWBN. The following steps run in \( O(1) \) time using \( O(N) \) processors with \( O(N) \)-row and \( O(N) \)-column bus networks.

**Step 6.1** //for each vertex \( u \), find the set of children (descendants)/

Processor \( P_i, 0 \leq i \leq 2N-2 \), holds the element \( A(i) \) of array \( A \) with \( A(i)=k, 0 \leq k \leq N-1 \), using the \( k \)-th row bus network to find the processor with the minimal and maximal indices by Lemma 7.

**Step 6.2** Processor \( P_i, 0 \leq i \leq 2N-2 \), with the maximal index broadcasts the element \( B(i) \) of array \( B \) to processor \( P_j, 0 \leq j \leq 2N-2 \), with the minimal index using the \( k \)-th row bus network, if \( A(i)=k \) and \( A(j)=k \).

**Step 6.3** Processor \( P_i, 0 \leq i \leq 2N-2 \), with the minimal index computes \( \text{ pst}(k)=B(i)-B(j) \), if \( A(j)=k \).

**Step 6.4** Processor \( P_j, 0 \leq j \leq 2N-2 \), with the minimal index broadcasts \( \text{ pst}(k) \) to processor \( P_k \) using the \( k \)-th row bus network, if \( A(j)=k \).

**Theorem 5**
Given a tree \( T=(V, E) \) rooted at \( r \) with \( N \) vertices, which is represented by a set of edges, the tree recursions problem can be solved in \( O(1) \) time on an RAPWBN using \( O(N^{1+1/5}) \) processors with \( O(N) \)-row and \( O(N^{1+1/5}) \)-column bus networks.

**Proof:**
The correctness of the algorithm follows from Lemma 8. The time complexity is analysed as follows. Step 1 takes \( O(1) \) time using \( O(N) \) processors by Theorem 3. Steps 2 and 3 take \( O(1) \) time using \( O(N) \) processors by Theorem 4. Step 4 takes \( O(1) \) time using \( O(N) \) processors. Step 5 takes \( O(1) \) time using \( O(N^{1+1/5}) \) processors by Theorem 2; Step 6.1 takes \( O(1) \) time using \( O(N) \) processors by Theorem 7. Steps 6.2, 6.3 and 6.4 take \( O(1) \) time using \( O(N) \) processors, respectively. Hence, the time complexity is \( O(1) \).

In addition to the prefix sum on a tree, there are many other interesting problems on a tree that can be solved in constant time. We list them as follows.

1. Compute the number of descendants of each vertex.
2. Compute the depth of each vertex of a tree.
3. Compute the height of a tree.
4. Compute the path length (or external path length) of a tree.

Problem (1) can be solved by taking \( \text{val}_v=1 \) for each vertex \( v \). Assume the depth of the root of a tree is 0. Since the depth of each vertex is equal to its transitivity, problem (2) can be solved by the list ranking algorithm. After the depth of each vertex of a tree is computed, problem (3) can be solved by finding the maximum depth of a tree. The minimum (or maximum) of \( N \) data can be determined in constant time by Lemma 4. Assume each vertex has been associated with a degree. Problem (4) can be computed by first computing the depth of each vertex. Then identify the pendant vertices (a pendant vertex was defined as a vertex of a degree of one). Finally, those depths of pendant vertices can be summed by Corollary 4.

**5 CONCLUSIONS**
In this paper, two basic operations for computing the prefix modular \( n \) and prefix division \( n \) of an \( N \)-bit binary sequence are first proposed in \( O(1) \) time on a linear RAPWBN using \( N \) processors. Then based on these operations, many related problems such as those discussed in this paper can be solved efficiently. Using the results derived in this paper, it is possible that many related problems can be solved efficiently in the near future.

Owing to the local switches established between the system buses and the processors, the architecture of any reconfigurable parallel processing system can be reconfigured at run time. Some researchers may offer the criticism that there is a connection delay for the local switch. We agree that the connection delay will depend on the problem size so that the constant time for broadcasting delay is not true. However, although it is not true, the broadcasting delay is very small. It is still practical as there are two VLSI implementations that
have demonstrated the feasibility and benefits of reconfigurable parallel processing systems: one is the YUPPIE (Yorktown Ultra-Parallel Polymorphic Image Engine) chip proposed by Li and Maresca (1989b) and the other is the GCN (Gated-Connection Network) chip proposed by Shu and Nash (1988). Li and Maresca (1989b) pointed out the switching delay for a local switch was no more than 1 ns. For example, in a 10^6 processors YUPPIE, only 16 machine cycles are enough for broadcasting. Using the pre-charged circuits, the GCN has further reduced the delay. In a practical implementation, the line delay is sometimes longer than the connection delay. Indeed, we are not concerned about how much time the broadcasting delay will be as it fully depends on the physical implementation technique. The concept of a reconfigurable system is worth emphasizing. We believe that such a configuration may be becoming the architecture of the next generation of parallel computers.

REFERENCES


ROTHSTEIN, J. (1976): On the Ultimate Limitations of Parallel Processing, in Proc. of Int. Conf. on Parallel Processing, 206-212.


STOUT, Q.F. (1985): Tree-Base Graph Algorithms for Some Parallel Computers, in Proc. of Int. Conf. on Parallel Processing, 727-731.


BIBLIOGRAPHICAL NOTES

Tzong-Wann Kao received his B.Sc. in Computer Science from Tamkang University, his M.S. in Electronic Engineering from the National Taiwan Institute of Technology, and his Ph.D. in Electrical Engineering from the National Taiwan Institute of Technology in 1981, 1983 and 1994, respectively. Currently, he is an Associate Professor in the Department of Electronic Engineering, Kuang Wu Institute of Technology and Commerce. His research interests include image processing and parallel algorithms.

Shi-Jinn Horng received his B.Sc. in Electronic Engineering from the National Taiwan Institute of Technology, his M.S. in Information Engineering from National Central University, and his Ph.D. in Computer Science from National Tsing Hua University in 1980, 1984 and 1989, respectively. Currently, he is a Professor in the Department of Electrical Engineering, National Taiwan Institute of Technology. His research interests include VLSI design, multiprocessing systems and parallel algorithms.
NEWS BRIEFS

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

WORLD PC BREAKTHROUGH BY AUSTRALIAN R&D GROUP
Stone Group Asia Pacific Investments Limited, through its wholly owned R&D company Stone Microsystems (Australia) Pty Ltd, has developed the world’s first multi-video hardware which allows up to 16 terminals to run multiple applications off a single central PC.

The Stone Multi-Video VGA system will have widespread applications for banking and finance, office automation, education, SMEs, point of sale, professional practices and network access.

Under an agreement with leading PC manufacturer, AST Computer, the Stone Multi-Video VGA system will be released to the US market and subsequently worldwide.

Stone Group Managing Director, Mr Yin Ke said: “This is the start of a revolution in the PC marketplace. It is an Australian technology breakthrough which has massive global potential. Our agreement with AST Computer to release this product to the market is a vindication of our decision to concentrate on R&D”.

The Multi-Video VGA system will eliminate the need and cost of a separate CPU at each desk and will lower the cost of computing by reducing the initial investment and limiting support and upgrade costs to only one system.

Stone's Multi-Video VGA system enables the connection of up to 16 terminals, each consisting of a colour monitor, keyboard and mouse, to one PC.

AST Computer has incorporated it in Centrala, a new PC product which is the world's only Window-based, multiple-user dedicated file server.

The terminals provide a user-friendly multi-tasking environment supporting colour graphics applications and features previously only available on a single PC. The Stone technology now provides a low cost, high performing, multi-user PC network.

The Multi-Video VGA system is especially suitable for Asian markets where the local languages require graphics display capabilities for every user. The system provides a cost-effective alternative to Local Area Networks (LAN) with simpler maintenance and improved integration.

TAFE QUEENSLAND'S NEW APPROACH TO INFO TECH QUALIFICATIONS
TAFE Queensland has just completed development of a range of new Info Tech qualifications.

The new courses bring a significant increase in the range of specialist training available to the IT industry in Queensland. Mr Col Chandler, Queensland Manager of JNA Networks, and advisor to TAFE Queensland curriculum developers, explained that "A number of features should make the TAFE Queensland courses particularly attractive to IT enterprises and employees looking to maintain and diversify their skill base".

Mr Chandler sees the development of training opportunities such as this one as critical to the future of the IT Industry in Queensland.

Mr Sherman Sawtelle, project manager for TAFE Queensland, indicated that "The new courses start from Certificate II (basic computer literacy) and cover every level up to Advanced Diploma (professional analysts, programmers etc.). The development brings the first IT credential available at the new Advanced Diploma level, and an increase from 5 specialised courses to 18 new specialised awards available from TAFE Queensland".

Mr Sawtelle explained that "the Certificate IV can be completed part-time over one semester (or 200 hours by flexible mode) On completion, depending on their choice of strand, students will be able to undertake 3GL, 4GL and/or Object orientated programming, systems analysis, and PC and Network support. These graduates would be of immediate business use as trainee programmer/trainee support officers".

Prospective students can check the course details in the latest TAFE Queensland Handbook (full-time enrolments into Diploma or higher level must apply through QTAC). Alternatively they may wish to contact their local TAFE college.

AUSTRALIAN COMPUTER EXPERTISE ATTRACTS MAJOR OVERSEAS SALE

The National Film and Sound Archive in partnership with the private sector has developed a unique computer software package which has attracted great interest from audiovisual institutions overseas. The software has been sold to the Norwegian Government, and there is strong interest from institutions in North America and Asia.

The Merged Audio Visual Information System, known as MAVIS, has been developed in partnership with the Canberra-based computer company Wizard Information Services. A collection management system, MAVIS is a sophisticated suite of software which facilitates the management of large collections of audio visual material.

The Minister for Communications and the Arts, Senator Richard Alston, applauded the development of MAVIS saying that it is an excellent example of Australian creative expertise with proven export potential. (The Archive is a semi-autonomous agency of Department of Communications and the Arts).

The MAVIS system is a unique product that manages audiovisual information from acquisition through to conservation and preservation. It not only describes the items held in the collection, it identifies exact location, physical condition and how to retrieve the items.

Nothing like it exists anywhere else in the world.

"The beauty of MAVIS is that it provides greater ease of access to over one million items held in the National Collection", said the Archive’s Director, Ron Brent. "Cutting edge technology has been employed in cultural heritage management and we are very pleased with the result".

ELECTRONIC ADS HIT THE INTERNET
Yellow Pages Australia — the first Yellow Pages® publisher in the world on the Internet — has announced an electronic advertisements (interactive web ads) trial.

The trial is part of the Company's strategy to quantify the commercial viability of advertising on the Internet and determine user preferences.

Yellow Pages Australia's National Multimedia Manager, Mr Bob Copp said: "Advertisers are thinking more strategically about advertising on the 'Net', so we want to demonstrate to customers usage and value in trial mode before any commercial launch".

"While advertising revenues on the World Wide Web total about US$12 million a month,1 Australian businesses are yet to be convinced of the economic return from advertising on the 'Net', he added.

Selected Yellow Pages customers advertising in the 'Computer Equipment Hardware' and Motor Cars New classifications in the Sydney and Melbourne directories have participated in the trial, with 50 interactive web ads on the site today, growing to about 100 by the end of September.

Located within the popular Australian Yellow Pages site, now also a local Australia™ affiliate site (http://www.aluvista.yellowpages. com.au), the advertisements feature photographs, graphics and text specifically designed to suit Internet functionality and performance.

The interactive web ads have already attracted users who have contacted the Companies via e-mail with purchase enquiries.

The trial is part of Yellow Pages Australia's plan to stay at the forefront of technology-based business directory publishing. The Company was also the first in Australia to put digitised advertisements on the Internet in June this year.

Mr Copp said no decision would be taken about the full commercialisation of electronic advertising until the completion of the trial later this year.


AUSTRALIA AND NEW ZEALAND DEFENCE ON DISC (ANZ-DoD)

Australia and New Zealand Defence on Disc (ANZ-DoD) was released recently at the Defence Procurement '96 conference in Canberra.

Now in its third issue, ANZ-DoD is the one-stop-shop for defence procurement information, containing a wide range of company, defence department and government information.
world first concept developed by IHS Australia and distributed throughout Australia, New Zealand and the World.

ANZ-DOD is used by the defence industry, defence department buyers around Australia and New Zealand, defence attaches in Australian embassies, Aus Trade offices, and top advisors in key defence export markets in Asia, North America and Europe.

ANZ-DoD uses a unique combination of Fox Pro database and Adobe Acrobat software that allows fielded free text searching, video clips, animations, slide shows, simulations. It can also accept full colour reproduction of company promotional material, annual reports, catalogues and colour advertisements.

Users can search for information by a range of defence department classifications as well as by standard industry code, ACN or by keyword. This unique CD-ROM allows defense industry to present their products, services and capability statements in a simple, accessible and intuitive way.

For more information about ANZ-DoD please call Graham Manns or George Mousa at IHS Australia on 1 800 803 958.

PROTECTION FOR THE ELECTRICAL "SUPERHIGHWAY"

Surge protecting switched powerboard.

HPM Industries has developed a new range of Switched Powerboards, including a Surge Protected Switched Powerboard.

The new surge protected version contains an MOV (metal oxide varistor) which absorbs extra voltage from a power surge, leaving a steady stream of regulated power flowing to the equipment.

The neon built into the plug top on the powerboard also provides peace of mind as a "glowing" assurance that the unit is still protecting your equipment.

With HPM’s Surge Protected Switched Powerboard, never again will you experience that hide and seek sensation while wrapped in a web of cords under the desk tagging at plugs trying to turn off the modem. Taking identification one step further, this model comes with labelled inserts that slide into the switches, allowing each of the four outlets to be correctly and quickly identified. These include commonly used home and office appliances, such as PC, printer, fax, modem, TV, video and stereo.

For further information contact HPM Industries Customer Service on toll free 1 800 80 7000.

AUSTRALIA COUNCIL CONNECTS ARTISTS WITH NEW TECHNOLOGY

The Australia Council has announced two new residencies for Australian artists to work with enterprising, innovative organisations specialising in digital technology.

Valued at $35,000 per annum, the new residencies are with Firewire, one of Australia’s leading digital media groups, and the CSIRO Division of Information Technology.

The residencies will give artists access to a high technology environment and expertise, training in new technologies, new opportunities to distribute and publish their work and a further understanding of the technical and conceptual issues in new media. They offer artists the chance to apply their ideas and creative solutions to industry.

Artists will be involved with special projects and ongoing research as an integral project member of an information team, investigating and applying these new technologies. Artists will also be able to pursue their own work using the facilities of CSIRO and Firewire.

CSIRO’s Division of Information Technology is actively promoting research on the convergence of information technology, communications, electronic publishing, film, arts and design. The Division is working with advanced technologies for virtual reality, visualisation, on-line information services, artificial intelligence, high performance computing and multimedia.

“The Australia Council welcomes this substantial financial and technical support from industry. The convergence of the arts and technology through the collaborative exchange of information and skills will open up an exciting creative dialogue”, said Michael Lynch, General Manager of the Australia Council.

“The convergence of the computing, communications and content industries is creating an opportunity for innovation from multi-disciplinary teams. We see the residency scheme as fostering these teams and nurturing innovation in the Information Age”, said Dr John O’Callaghan, Chief of the Division of Information Technology.

The closing date for these art and technology residency applications is 15 November, 1996.

Artists can obtain more details, the Australia Council Grants Handbook and application forms from the New Media Arts Fund, Australia Council, PO Box 788 Strawberry Hills NSW 2012. Tel (02) 9950 9000. Toll free 1800 226 912. Fax (02) 9950 9111. Email newmedia@ozco.gov.au.http://www.ozco.gov.au.

DESKTOP PROJECTORS

Proxima Corporation has released a new range of desktop projector products including the MediaExpress, which incorporates the Adobe Acrobat Portable Document Format (PDF) file technology.

For further information contact HPM Industries Customer Service on toll free 1 800 80 7000.

Distributed by B&H Australia, the Proxima desktop projector range now includes four new multimedia projectors with differing capabilities of resolution, brightness, price and solutions for customer applications.

The Proxima Desktop Projector DP100z boasts a new option known as Media Express. This option enables most computer generated presentations including spreadsheets, graphics, word documents and PowerPoint images to be projected from a 3.5 inch disk inserted directly into the projector.

“This remarkable new technology means that presenters no longer need to worry about computer and cable connections, or indeed computer platforms”.

“Once their presentation is created on either a PC, Macintosh or a workstation computer, it is a simple matter to save the presentation as an Adobe Acrobat file direct to the disk for insertion into the Desktop Projector equipped with MediaExpress”, explained Barry Smith, Sales Director of B&H.

MediaExpress includes Adobe Acrobat software, the cross-platform standard for sharing electronic information, plus Proxima unique plugs for reducing file size and improving on screen presentation effects.

The DP100z also incorporates the latest Digital Light Processing (DLP) technology from Texas Instruments Corporation.

“Featuring hundreds of thousands of digital light switch micromirrors to reflect beams of light, the DP4100z projects extremely bright photorealistic images — even in a very well lit room”, Smith said.

Also in the recently released range is the industry’s brightest multimedia projector, the DP2910, projecting an astounding 650+ ANSI lumens — unmatched in image brightness and clarity, and the DP5500, for rooms requiring a long-throw lens and an image of up to 9 metres diagonally.

For users of high resolution PCs and workstations, the DP9100 XGA Desktop Projector projects large screen images up to 1280 x 1024 in 24 bit colour. MediaExpress is also an option with the DP9100 projector.

B&H Australia (formerly Bell & Howell Audio Visual) are the exclusive distributor for Proxima Corporation of San Diego, in Australia and New Zealand, and offer an extensive range of Proxima Desktop Projectors and panels for large screen computer and video projection.