計算機アーキテクチャ　106－8
（1994．6．13）

レジスタ挿入バス方式による高速相互結合網

Andrew C. Flavell, 高橋 義造
徳島大学工学部知能情報工学科
徳島市南常三島町2-1
{flavell,taka}@is.tokushima-u.ac.jp

相互結合網に関する多くの研究が行われているが，これは並列計算機の性能を左右する重要な要素であり，さらなる研究が必要である．本論文では調停時間とスケーラビリティに問題のある時分割バスを改良し，複数の単方向レジスタ挿入バスを用いてアダプティブルーティングを行う時間空間分割型ハイブリッド相互結合網を提案する．この結合網の性能をシミュレーションにより評価した結果，マルチプロセッサ用相互結合網として勝れた性質を持つことが確かめられた．

A High Speed Register Insertion Bus Multicomputer Interconnection Network

Andrew C. Flavell and Yoshizo Takahashi
{flavell,taka}@is.tokushima-u.ac.jp

Department of Information Science and Intelligent Systems,
Faculty of Engineering,
University of Tokushima.

2-1 Minami Josanjima-cho, Tokushima 770,
JAPAN

Although major advances have been made in improving the performance of interconnection networks for parallel multiprocessor systems, this area continues to be an active avenue of research due to the important role that the interconnection performance plays in determining the overall performance of a parallel computer system. In this paper we introduce the register insertion bus interconnection network which utilizes multiple, unidirectional, register insertion buses to provide a hybrd time/space division network. This structure overcomes the problems of slow arbitration speed and scalability of time division networks, while retaining their desirable characteristics. We also apply shortest queue output selection along with adaptive routing to more effectively utilize the bandwith of the interconnection network. The register insertion bus interconnection network is evaluated by simulation and is found to provide an effective model for the implementation of a multiprocessor interconnection network.

Keywords: multi-processor, interconnection network, space/time division commnunication, k-ary n-cube

1. INTRODUCTION

There are many possible topologies for the interconnection networks (INs) employed in massively parallel computer (MPC) systems, but by far the most popular in the current generation of MPCs are the family of *k*-ary *n*-cubes and those networks which are isomorphic to them.

The 2D torus network is utilized in the iWarp[1] and K2 parallel processor[2], the 3D torus in the Cray Research T3D[3], the 2D mesh in the Intel Paragon and the MasPar MP-1[4], and a 3D mesh has been utilized in the J-machine[5]. Most of these systems utilize space division or direct interconnection networks, even though many optimum algorithms exist for *k*-ary *n*-cubes utilizing time division networks. These algorithms include the nearest neighbor problem[6], the calculation of the extreme points of a convex hull in a digitized image[8] and a number of linear algebra algorithms [8]. The preference of MPCs utilizing space division networks is primarily due to the problems involved in scaling time division networks because of the limits of arbitration speed, and poor performance under high-traffic conditions[9]. Thus, our motivation in the formulation of an IN using register-insertion buses (RIBs) was to support the communications model of time division networks, while overcoming the problems of slow arbitration speed and poor scalability inherent in them.

The remainder of this paper is organized as follows; We begin in Section 2 by introducing the RIB and then discuss the structure of a generic communications router suitable for the implementation of RIB INs. In Section 3 we demonstrate two algorithms which can take advantage of our IN structure and this is followed in Section 4 by an evaluation of the performance of RIB INs by simulation. Finally, a summary and proposals for further investigation are presented in Section 5.

2. RIB INTERCONNECTION

2.1 Register Insertion Bus

High performance ring buses have become a favorable alternative in the implementation of local area networks[10]. However, LAN/WAN structures are not directly applicable to INs due to differences in the node structure and communications patterns[11]. The use of the unidirectional register-insertion bus in the construction of INs does, however, have a number of advantages. With reference to Figure 1, which illustrates the structure of a bus interface for a register-insertion bus, these advantages include:

• A packet may propagate through a large number of bus interfaces without being buffered.
• Processors are free to inject packets at any time, subject to available space in the transmit queue. Thus there is no global arbitration, as each processor can decide whether to inject a packet according to information local to its bus interface.
• Active repeaters can be used at the output of each message router, instead of the pulldown structure required for a bi-directional bus, thus making the network more scalable.

The basic operation of a register-insertion bus is as follows; We assume that the input and output data is synchronized at the same transmission rate, so that for each word received, another can be transmitted. The transmit (tx.) buffer is used to temporarily store a packet from the local processor while it is waiting for injection onto the bus. These packets are of variable length and so only a portion of the tx. buffer may be used for a particular packet, however, the packet length must not exceed the length of the tx. buffer. The function of the delay buffer can explained by first considering the area currently being used. The used, or active portion of the delay buffer, operates as a FIFO queue that delays the incoming packets. Assuming that the entire delay buffer has a capacity of n words and that i words are currently used, $1 \le i \le n$, then $n-i$ words remain for the unused or inactive portion. Thus locations $w_0, w_1, \ldots, w_{i-1}$ of the delay buffer are active and locations $w_i, w_{i+1}, \ldots w_{n-1}$ are inactive. If, in each time step t, a new word can be received, and a new word is to arrive at time $t+i$, then the active portion of the delay buffer represents a FIFO queue containing the words which arrived at times $t, t+1, \ldots, t+(i-1)$. As the capacity of this FIFO queue is i words, at time $t+i$ the word stored in w_0 is removed from the queue and sent to the output. Simultaneously, the incoming word is added to the queue such that locations $w_0, w_1, \ldots, w_{i-1}$

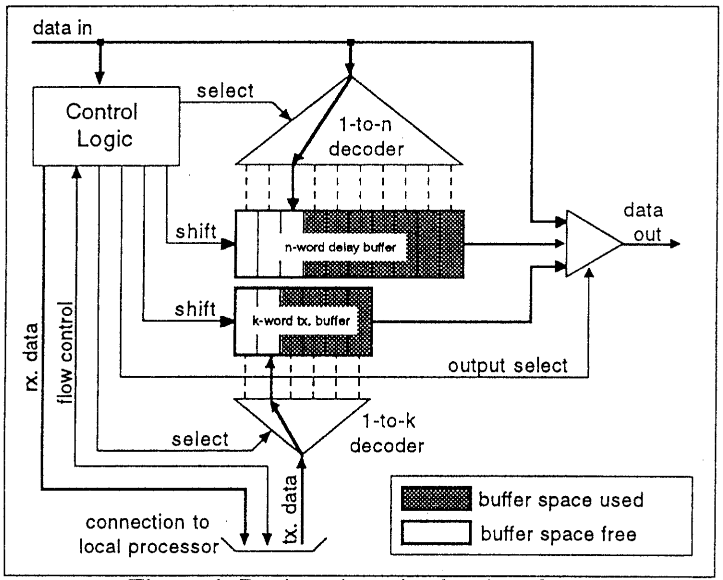

Figure 1. Register-insertion bus interface.

now contain data which arrived at times $t+1, t+2, ..., t+i$, and the queue length remains unchanged.

It is desirable that in each time step, if $i \geq 1$, the queue size be reduced. A reduction can take place *iff* the data received at the input is not part of any packet destined for the output. In this case, the previous discussion should be modified so that the incoming word is not stored in location w_{i-1} and also so that i is reduced to $i' = i-1$. Furthermore, if $i = 0$, then any incoming word need not be stored at all and can pass directly to the output. In this case the incoming word is not stored in location w_{i-1} and i is constant at $i = 0$.

The inactive portion of the buffer is essential for the injection of packets into the network from the tx. buffer. Assuming that the tx. buffer contains a packet of length l, $1 \leq l < (n-i)$, and that at time $t+i$ the first word of the of this packet is to be sent to the output, then the previous FIFO discussion should be modified as follows; In this case, at time $t+i$ the incoming word is stored in location w_i and i is increased to $i' = i+1$. At time $t+((i+l)-1)$, after the last word of the transmitted packet has been sent, the locations $w_0, w_1, ..., w_{(i+l)-1}$ of the delay buffer now contain words $t, t+1, ..., t+((i+l)-1)$. In addition, the requirements for queue reduction must be modified such queue that reduction can only occur *iff* the data received at the input is not part of any packet destined for the output and no packet is currently being sent from the local tx. buffer.

From the preceding discussion we can observe that if $i = 0$, the delay experienced by a packet is only due to the propagation delay through the output selector. Also if no packet is being sent from the transmit buffer and i is less than the length of the incoming packet, then the packet will cut-through the FIFO. Finally if i is greater than the length of the incoming packet, or a packet is being transmitted and l is greater than the length of the incoming packet, then the incoming packet will be completely buffered in the FIFO, in a store-and-forward manner.

2.2 Register-Insertion Bus k-ary n-cube INs

The concept of the register-insertion bus can easily be extended to the k-ary n-cube as is shown in Fig. 2, which illustrates the structure of one port of a generic message router for an n-dimensional register insertion network. The delay buffer of Figure 1 is replaced by a group of output buffers. These buffers store packets that are changing dimensions, in addition to those which must be delayed while the local processor injects new packets into the network. Also, the control is now distributed between the input and output control sections to improve performance.

With reference to Fig. 2, the operation of the router is as follows; All packets headers which appear at an input are processed by the corresponding input controller. Thus, a packet traversing dimension i will be processed by input control i. Digit i of the destination address $d_0, d_1, ...d_{n-1}$ in the header is compared to digit i of the current node's address $a_0, a_1, ...a_{n-1}$. If $d_i = a_i$, or if the output of the current dimension is sending, the packet will then request an new output. If $d_i \neq a_i$ and the output of current dimension is not sending,

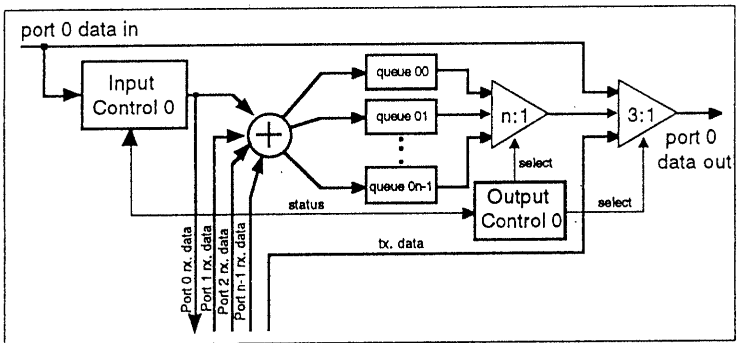

Figure 2. Port 0 of generic router for a register-insertion k-ary n-cube.

then the length of the packet is stored and a flag is set in the input controller to indicate that a packet is bypassing the current node and cannot be interrupted. The bypass counter is decremented each time a valid word of the bypassing packet is detected, until it reaches zero.

Each output queue may accept data from multiple sources in a single clock cycle and a packet which is $(d \geq 2)$-dimensions from its destination will sequentially request those outputs which will enable it to be routed closer to its destination, in order of increasing output queue length. If the output granted to a packet is currently free, the packet may be immediately re-injected into the network, and thus the network implements cut-through routing. If a packet is not granted an output which allows it to progress towards its destination, it is misrouted to a randomly selected channel. The use of misrouting ensures that an input packet will always be routed and thus prevents deadlock from occurring, while the introduction of randomness into the selection of the output to which the packet is to be misrouted has been shown by Konstantinidou and Snyder [13] to ensure that a network is probablistically free from livelock.

In the absence of blocking, a unicast packet in a d-dimensional network may only be temporarily stored in the d-nodes where the packet must change dimensions, even though all of the intermediate nodes in the path of the packet will detect its presence. This highlights two interesting properties of the RIB. First a unicast in a single dimension can be viewed as a multicast, in which all of the nodes in the message path except intermediate or destination nodes discard the message. We can take advantage of this multicast-like-unicast in the implementation of a broadcasting scheme. Second, in the absence of blocking there is a virtual direct connection between any two nodes which lie on the same bus, as is the case with a time division bus.

3. ALGORITHM MAPPING

In this section we present two examples of classes of algorithms which can take advantage of the communications structures of the RIB paradigm.

3.1 Recursive Doubling

Consider taking the sum of $N = n \times m$ numbers, distributed uniformly over an 2-dimensional Continuum network, with N processing elements. Embedding an n-level tree into a Continum network and utlilizing the virtual direct-connection between processors, enables this to be computed in $O(\log_2 N)$ time, by performing $\log_2 N$ transfer-adds. A space-shared mesh network can perform the same operation in $O(n)$ time. The summing tree for eight numbers is presented in Fig. 3(a) and the mapping of

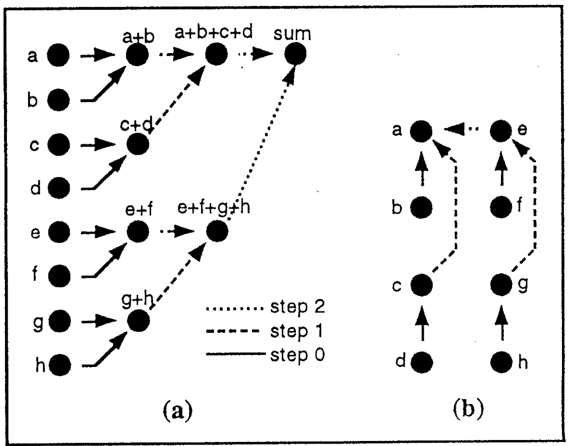

Figure 3.(a) 'Tree summing' of eight numbers. **(b)** Mapping of tree to Continuum network

this tree to a 4×2 Continuum network is presented in Fig. 3(b)

3.2 Matrix Operations

The use of broadcasting has been identified as the key to achieving optimal speedup for the solution of a number of parallel algorithms. A large number of these algorithms can be expressed in terms of matrix multiplication operations, which can be described as follows:

$$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj} \qquad i,j = 1,2,\ldots\ldots,N$$

where A,B and C are assumed to be square matrices. It can be seen that all of the elements of a given row in A and column in B will contribute to an element in the result matrix C, and thus broadcasting, or multicasting, can be used to perform efficient matrix multiplication.

Matrix inversion by the Gauss-Jordan triangulaization method, in which the source matrix A is progressively transformed into an identity matrix by subtracting each row, appropriately weighted, from the other rows, and the identity matrix I is transformed into A^{-1} by applying the same sequences of transformations which are applied to A., to I, can benefit significantly from broadcasting. Given a 2-dimensional Continuum network, with $N = n \times n$ nodes, and the $n \times n$ matrices A and I distributed uniformly over the network such that node n_{ij} holds elements a_{ij} and i_{ij} the inversion is performed as follows: At each iteration k all of the elements of a_{ij} of A , where $i,j = 0,1,\ldots,n-1, i \neq k$, are updated to $a_{ij} \leftarrow a_{ij} - (a_{ik}/a_{kk}) \times a_{kj}$ and similarly, all of the elements i_{ij} of I, where $i,j = 0,1,\ldots,n-1, i \neq k$, are updated to $i_{ij} \leftarrow i_{ij} - (a_{ik}/a_{kk}) \times a_{kj}$. In order to perform each update, each node n_{ij} requires a_{kj}, a_{ik} and a_{kk}; column a_{kj}, which includes a_{kk}, is multicast in the +X (-X) direction, followed by a multicast of a_{ik}/a_{kk} in the +Y (-Y) direction.

4. SIMULATION RESULTS

Although the register-insertion bus may be extended for use in any k-ary n-cube, two or three-dimensional networks are generally considered optimum for implementation of INs for parallel systems[12]. To evaluate the performance of the RIB INs, we therefore evaluated several 8-ary 2-cubes, by simulation. The networks evaluated were a unidirectional torus, a bi-directional torus and a bi-directional mesh.

The simulator is a 5000 line C++ program with a graphical user interface and includes a dynamic display of the simulation progress. The display has proved invaluable in the development of the simulator, as well as providing insight into the results obtained. The simulator supports programmable network size, buffer size, routing algorithm, traffic pattern and packet length, as well as a test mode which can be used to check routing algorithms, buffer assignments etc. All of the nodes of the simulator operate synchronously and a word is transferred between nodes in a single clock cycle.

4.1 Simulation of Uniform Random Traffic

In order to measure the latency of each network, a constant rate source with exponential interarrival times was applied to each input, and the time from the creation of the first word of the packet until the last word of the packet is accepted at the destination was measured. Each simulation was run for 100,000 clock cycles and each node generated approximately 6000 packets.

Figure 4 compares the latency of the three networks for a packet size of 16 words and output queue length of 8 packets per port. The unidirectional torus and bi-directional mesh networks reached saturation at approximately 23% capacity and 48% capacity respectively, while the bi-directional torus did not reach saturation. The unidirectional torus network saturates quickly as each packet which is misrouted must completely traverse one dimension of the network before it is eligible to be profitably routed once more and thus the packet will remain in the network, occupying queue space and exacerbating the congestion. The non-symmetric structure of the mesh network results in higher loads and therefore higher congestion of the communications channels at the center of the network. Thus the congestion begins in the center of the network, and gradually propagates outward. The bi-directional torus network did not reach saturation as it

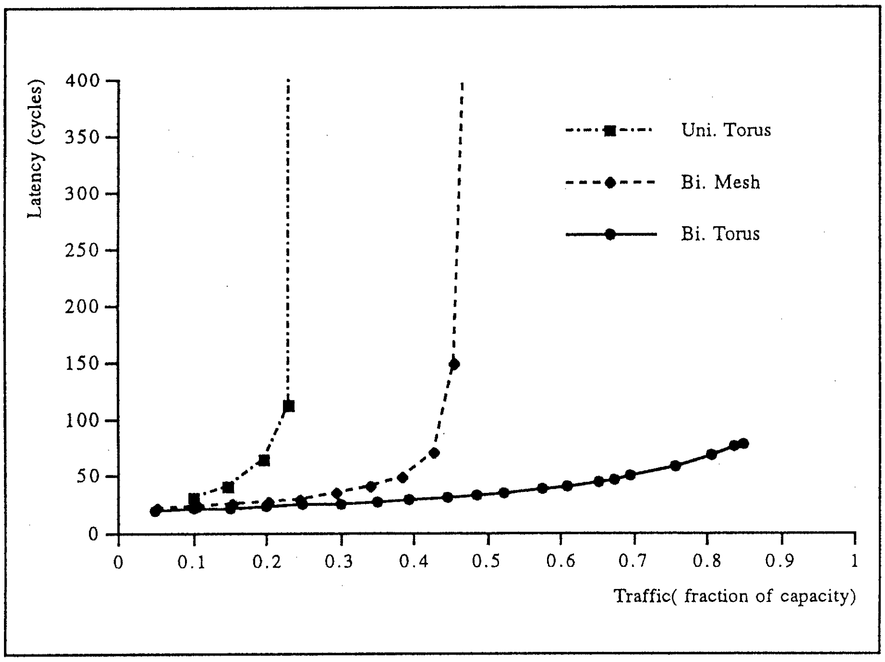

Figure 4. Latency vs traffic for unidirectional torus, bi-directional torus and bi-directional mesh.

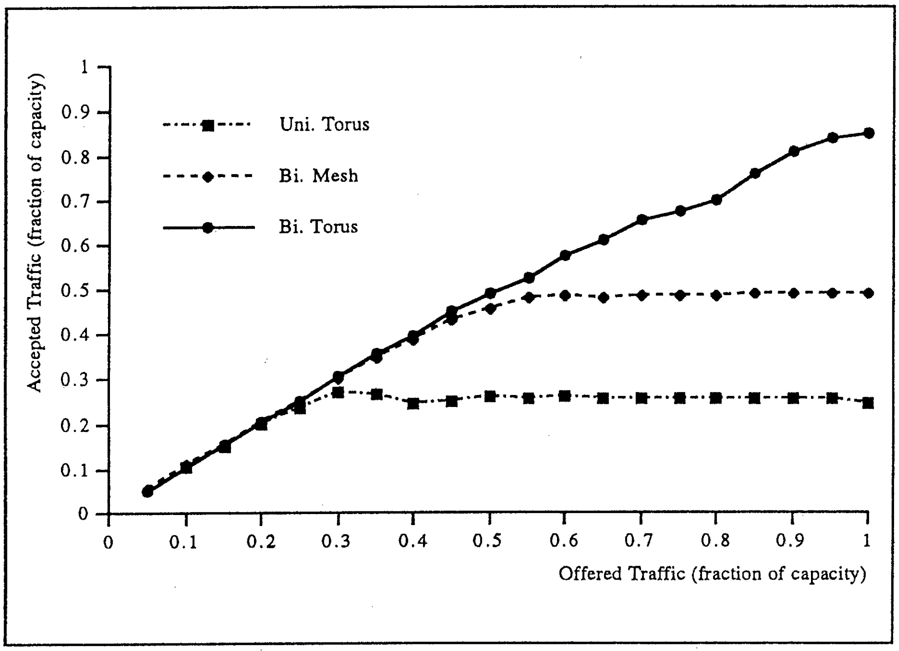

Figure 5. Throughput vs offered traffic for unidirectional torus, bi-directional torus and bi-directional mesh.

became source limited at approximately 83% capacity, i.e. the single port connection to the network of each node prevented them from injecting messages at a higher rate.

Figure. 5 compares the throughput of each network, as a function of offered traffic. The unidirectional torus achieves a peak throughput of 27% before stabilizing at a throughput of 23%. The bi-directional mesh reaches a stable state at approximately 48% throughput, while the throughput of the bi-directional torus increases approximately linearly until the network becomes source limited at 83%.

5. SUMMARY

We have examined the application of the RIB to the implementation of interconnection networks for massively parallel computers. The RIB IN utilizes multiple unidirectional, register-insertion rings to provide a hybrid time/space division network as a solution to the problems of slow arbitration speed and scalability of time division networks, and shortest queue output selection along with adaptive routing to more effectively utilize the bandwidth of the IN. Although register-insertion rings and buses have been successfully applied to LANs, we believe that this is the first such application of them to distributed memory multiprocessor interconnection networks.

The simulation results for the bi-directional 2D torus network are very promising and lead us to believe that the RIB approach can provide a useful tool for constructing interconnection networks. Clearly further investigation is required into the optimum structure of a communications router to support RIB INs, along with further evaluation by simulation of networks utilizing RIBs when subjected to communications patterns of applications programs.

References

[1] S. Borkar et. al., "Supporting Systolic Memory Communication in iWarp", *Proc. of the 17th Ann. Int. Symp. on Computer Architecture*, (May, 1990), pp. 70-81.

[2] M. Annaratone et. al., "The K2 Parallel Processor: Architecture and Hardware Implementation", *Proc. of the 17th Ann. Int. Symp. on Computer Architecture*, (May, 1990), pp. 92-101.

[3] W. Oed and M. Walker, "An Overview of Cray Research Computers including the Y-MP/C90 and the new MPP T3D", *Proc. of the 5th Ann. ACM Symp. on Parallel Algorithms and Arch*, (June, 1991), pp. 271-272.

[4] G. Zorpette, "Supercomputers/ Reinventing the Machine - The Power of Parallelism", *IEEE Spectrum* , (September, 1992), vol. 29. no. 9, pp. 28-33.

[5] W. J. Dally et. al., "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms", *IEEE Micro*, (April, 1992), pp. 23-39.

[6] V. K. Pransanna Kumar and C. S. Raghavendra, "Array Processor with Multiple Broadcasting", *Proc. of the 12th Ann. Int. Symp. on Computer Architecture*, (June, 1985), pp. 2-10.

[7] R. Miller et.al., "Parallèl Computations on Reconfigurable Meshes",*IEEE Trans. on Comp.ters*, (June, 1993), vol. 42, no. 6, pp.678-692.

[8] S. L Johnsson and Ching-Tein Ho, "Optimal Broadcasting and Personalized Communications in Hypercubes", *IEEE Trans. on Computers.*, (September, 1989), vol. 38, no. 9, pp. 1249-1268.

[9] K. C. Lee, "A Virtual Bus Approach for Dynamic Parallel Processing", *IEEE Trans. on Parallel and Distributed Systems*, (February, 1993), vol. 4, no. 2, pp. 121-130.

[10] C. C. Reames and M. T. Lui, "A Loop Network for Simultaneous Transmission of Variable-Length Messages", *Proc. of the 2nd Ann. Int. Symp. on Computer Architecture*, (January, 1975), pp. 7-12.

[11] W. J. Dally and H. Aoiki, "Deadlock-Free Adaptive Routing in Multicomputer Networks using Virtual Channels", *IEEE Trans. on Parallel*

and Distributed Systems, (April, 1993), vol. 4, no. 4, pp. 466-475.

[12] P. M. B. Vitany, "Locality, Communication, and Interconnect Length in Multicomputers", *SIAM Journal of Computing*, (August, 1988), vol. 17, no. 4, pp. 659-672, .

[13] S. Konstantinidou and L. Snyder, "Chaos router: Architecture and Performance", *SIGARCH*, (March, 1991), vol. 19, no. 1, pp. 212-221