Regular Paper

An Intelligent Policing-Routing Mechanism Based on Fuzzy Logic and Genetic Algorithms and Its Performance Evaluation

Leonard Barolli,† Akio Koyama,†† Takako Yamada† and Shoichi Yokoyama†††

The Asynchronous Transfer Mode (ATM) has been standardized and widely accepted as a technique to support future B-ISDN networks. Two of important functions for traffic control in ATM networks are policing and routing. All previous studies have treated policing and routing in a separate way. A combination of policing and routing can guarantee a better quality of service and network utilization. So far, many network control strategies have been proposed, but they are not well suited for high speed networks. To cope with rapidly changing network conditions, the traffic control methods for high speed networks must be adaptive, flexible, and intelligent for efficient network management. Use of intelligent algorithms based on fuzzy logic, neural networks and genetic algorithms can prove to be efficient for traffic control in ATM networks. In this paper, we propose an intelligent policing-routing mechanism which is based on fuzzy logic and genetic algorithms. Performance evaluation via simulations shows that, the proposed mechanism performs better than conventional policing mechanisms and routing algorithms.

1. Introduction

The Asynchronous Transfer Mode (ATM) networks has been standardized and widely accepted as a technique to support future B-ISDN networks. In ATM networks, the traffic control design becomes an important challenge, because of the diverse services support and the need for an efficient network resource engineering. Two of important functions for traffic control of ATM networks are Policing Mechanisms (PMs) and Routing Algorithms (RAs).

The purpose of PMs is to act on each source before all the traffic is multiplexed, in order to guarantee the negotiated Quality of Service (QoS). The proposed parameters for source policing are the mean cell rate, the peak cell rate or the peak burst duration [1),2)]. Policing of the peak cell rate is generally not complex and can be achieved by using a cell spacer or other PMs [3)]. Monitoring of the mean cell rate is more difficult, but is intended to improve the link utilization when it has to handle bursty traffic sources. The conventional PMs proposed in Refs. 1)–4), can't efficiently monitor the mean cell rate of bursty sources. The Window Mechanisms (WMs) are not well suited to the real-time services of the speed envisaged for B-ISDN, and the Leaky Bucket Mechanism (LBM) in the case of mean cell rate control requires a very high counter threshold to obtain an acceptable cell loss probability. This means that very long times are necessary to detect the violation of mean cell rate.

High speed transmission rates bring forward their specific issues influencing the RAs design. The RAs should give a fast decision to cope with traffic changes in ATM networks. The conventional Table-Based Routing (TBR) algorithms, which use routing tables, are computationally expensive and require a substantial amount of bookkeeping and periodic transmission of status information among the nodes. Also, the table size increases with the network size and can be large for a network with many nodes [5)].

The conventional PMs and RAs suffer from serious shortcomings. Some are simple but include many approximations and assumptions that are hard to justify. Others include complicated mathematical solutions that may not be feasible for real time implementation. Traditional network control strategies, which use queuing models for analytical evaluation, may not be effective because only the network steady state is assumed in queuing models. Therefore, to cope with rapidly changing network conditions, the network traffic methods must be adaptive, flexible, and intelligent for efficient

† Faculty of Literature and Social Sciences, Yamagata University
†† Faculty of Software Engineering, The University of Aizu
††† Faculty of Engineering, Yamagata University

network management.

Use of intelligent algorithms based on Fuzzy Logic (FL), Neural Networks (NNs) and Genetic Algorithms (GAs) can prove to be efficient for traffic control in high speed networks [6)~12)]. In Refs. 6) and 7), the FL is used to build a fuzzy policer, performance of which is better than conventional PMs and very close to ideal behavior of a PM. Bonde and Ghosh [8)] use a fuzzy threshold function for queue management in high-speed networks. The results show that, the FL provides a flexible and high performance solution to queue management in cell-switching networks. Cheng and Chang [9)] present a fuzzy traffic controller that simultaneously manages congestion control and call admission control in ATM networks. The fuzzy traffic controller is a fuzzy implementation of two-threshold congestion control method and the equivalent capacity admission control method. The fuzzy control improves system utilization by 11% compared with conventional method. In Ref. 10), a fuzzy controller for adaptive traffic in telephone networks is proposed. A simplified inference method is derived which attempts to represent gradual inference rules using fuzzy control. The inference method is based on heuristic rules derived from expert knowledge and human experience. It is proved that FL is an effective way to control the complex systems such as telecommunication networks. Some NN applications for traffic control in ATM networks are proposed in Ref. 11). The NNs are well suited to applications in the control of communications networks due to their adaptability and high speed. They can achieve an efficient adaptive control through the use of adaptive learning capabilities. The GAs are also a good approach for traffic control in high speed networks. A Genetic Load Balancing Routing (GLBR) algorithm is proposed in Ref. 12). The GLBR algorithm has a good performance compared with conventional Shortest Path First (SPF) and Route Information Protocol (RIP) algorithms.

In this paper, we propose an intelligent Policing-Routing Mechanism (PRM) based on FL and GAs. The proposed mechanism has three elements the Fuzzy Policing Mechanism (FPM), Tagging Switch (TS) and Tree based Adaptive Routing using GAs (TARG) algorithm. The performance evaluation via simulations show that the proposed PRM has a better behavior than traditional PMs and RAs.

The organization of this paper is as follows. In the next Section, we will give a brief introduction of FL and GAs. The source and network models will be treated in Section 3. The system model will be introduced in Section 4. The simulation results will be discussed in Section 5. Some implementation issues will be treated in Section 6. Finally, the conclusions will be given in Section 7.

2. Fuzzy Logic and Genetic Algorithms

2.1 Fuzzy Logic

The concept of a fuzzy set deals with the representation of classes whose boundaries are not determined. It uses a characteristic function, taking values usually in the interval [0, 1]. The fuzzy sets are used for representing linguistical labels. This can be viewed as expressing an uncertainty about the clear-cut meaning of the label. But important point is that the valuation set is supposed to be common to the various linguistic labels that are involved in the given problem [13)].

The fuzzy set theory uses the membership function to encode a preference among the possible interpretations of the corresponding label. A fuzzy set can be defined by examplification, ranking elements according to their typicality with respect to the concept underlying the fuzzy set [14)]. The prototypical element receives the greater membership grade. Fuzzy set naturally appears in non-strict specification. It may be soft constraints or flexible requirements for which slight violations can be tolerated (e.g., the dead line is today, but tomorrow is still acceptable although less good), or elastic classes of objects, approximate descriptions of types of situation to which a given procedure can be applied, or even procedures with fuzzy stated instructions.

The ability of fuzzy sets to model gradual properties or soft constraints whose satisfaction is matter of degree, as well as information pervaded with imprecision and uncertainty, makes them useful in a great variety of applications. The most popular area of application is fuzzy control. In fuzzy control systems, expert knowledge is encoded in the form of fuzzy rules, which describe recommended actions for different classes of situations represented by fuzzy sets. An interpolation mechanism provided by the fuzzy control methodology is then at work. A fuzzy control unit can do the same work as

a Proportional Integral Differential (PID) controller, since it implicitly defines a numerical function tying the control variables and the observed control variables together. However, by PID controllers only linear control laws can be attained, while the FL controller may capture non-liner laws, which may explain the success of the FL controllers over PID controllers. In fact, any kind of control law can be modeled by the FL control methodology, provided that this law is expressible in terms of "if ... then ..." rules, just like in the case of expert systems. However, FL diverges from the standard expert system approach by providing an interpolation mechanism from several rules. In the contents of complex processes, it may turn out to be more practical to get knowledge from an expert operator than to calculate an optimal control, due to modeling costs or because a model is out of reach [14].

2.2 Genetic Algorithms

GAs are search methods used to solve optimization problems. The GA mechanism is based on the interaction between individuals and the natural environment. GA comprises a set of individuals (population) and a set of biologically inspired operators (genetic operators). The individuals have genes which are the potential solutions for a problem. The genetic operators are crossover and mutation. GA generates a sequence of populations by using genetic operators among individuals. Only the most suited individuals in a population can survive and generate offsprings, thus transmitting their biological heredity to new generations [15].

GA operates through a simple cycle of four stages as is shown in **Fig. 1**. Each cycle produces a new generation of possible solutions for a given problem. At the first stage, an initial population of potential solutions is created as a starting point for the search. In the next stage, the performance (fitness) of each individual is evaluated with respect to the constraints imposed by the problem. Based on each individual's fitness, a selection mechanism chooses "parents" for the crossover and mutation operators. The crossover operator takes two chromosomes and swaps part of their genetic information to produce new chromosomes. The mutation operator introduces new genetic structures in the population by randomly modifying some of genes, helping the search algorithm to escape from local minima's traps. The offsprings produced by the genetic manipulation process

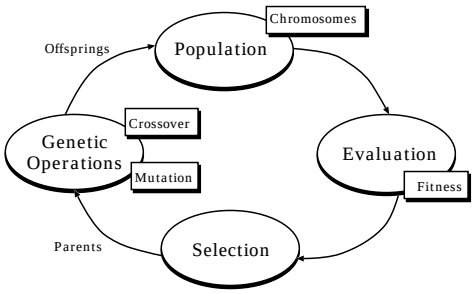

Fig. 1 GA cycle.

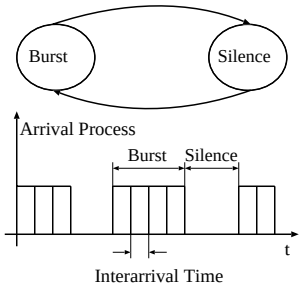

Fig. 2 Source model.

are the next populations to be evaluated. GA can replace either a whole population or its less fitted members only. The creation-evaluation-selection-manipulation cycle repeats until a satisfactory solution to the problem is found or some other termination criteria are met.

3. Source and Network Models

3.1 Source Model

We assume for the cell arrival process pattern a bursty source as shown in **Fig. 2**. Each burst has a duration *mbd* (mean burst duration) random variable and a cell rate of *pcr* cell/s (peak cell rate). The duration of inactive (silence) period is the random variable *msd* (mean silence duration).

The source is characterized by the following set of parameters: the peak (burst) cell rate [pcr]; the mean burst duration [mbd]; the mean silence duration [msd]; the source burstiness [$sb = (mbd + msd)/mbd$]; the mean burst length in cells (or burst cell number) [$bcn = pcr \cdot mbd$]; the mean source cell rate [$m = bcn/(mbd + msd)$]; and the mean cycle duration [$mcd = mbd + msd$].

3.2 Network Model

The network model with 20 nodes is shown in **Fig. 3**. In order to reduce the GA operation complexity, the network is transformed in a tree model. To explain this procedure, we consider a

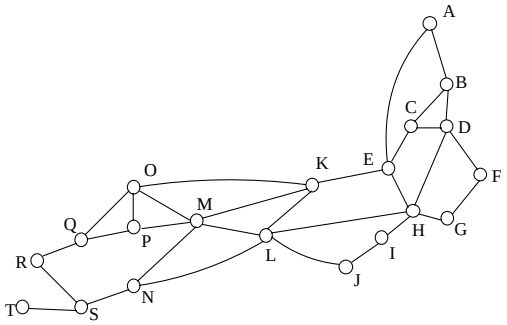

Fig. 3 Network model.

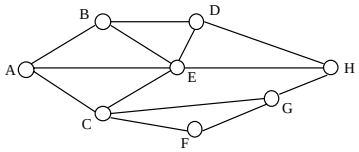

Fig. 4 Network example.

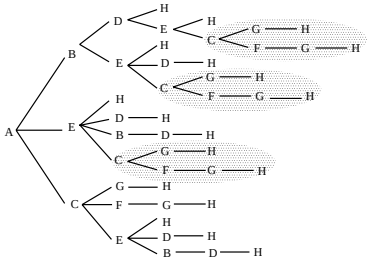

Fig. 5 Tree model.

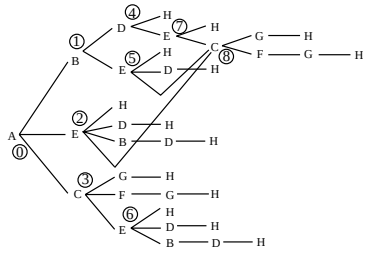

Fig. 6 Reduced tree model.

small network with 8 nodes as shown in **Fig. 4**. Node A is the source node and node H is the destination node. All paths are expressed by the tree model shown in **Fig. 5**. In the shaded areas are shown the same paths from node C to H. In order to decrease the chromosome gene number, the tree model of Fig. 5 is reduced as shown in **Fig. 6**. In the reduced tree model, each tree junction is considered as a gene and the path is represented by the chromosome (see Section 4.3).

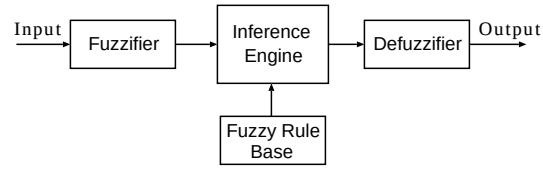

Fig. 7 FLC structure.

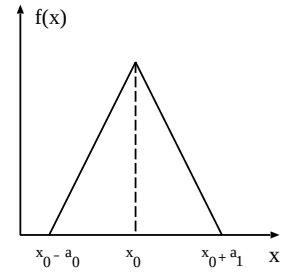

Fig. 8 Triangular membership function.

4. System Model

The system model has three elements: FPM, TS and TARG algorithm. They are described in following.

4.1 Fuzzy Policing Mechanism

The Fuzzy Logic Controller (FLC) is the major part of the FPM. The basic components of FLC are shown in **Fig. 7**. They are the fuzzifier, inference engine, Fuzzy Rule Base (FRB) and defuzzifier.

In the design of FLC, the triangular shape function is used because is easy to tune the membership functions. The function $f(x, x_0, a_0, a_1)$ for triangular shape is defined as follows (see **Fig. 8**):

$$f(x, x_0, a_0, a_1) = \begin{cases} \dfrac{x - x_0}{a_0} + 1 & x_0 - a_0 < x \le x_0 \\ \dfrac{x_0 - x}{a_1} + 1 & x_0 < x \le x_0 + a_1 \\ 0 & \text{otherwise} \end{cases}$$

where x_0 is the center of triangular function and a_j is the right/left width of the monotonic part of triangular function ($j = 0/1$).

In high speed networks, because of the unpredictable and often bursty real-time characteristics of traffic, network resources must be designed for average utilization. Therefore, the FPM is designed to control the mean cell rate of packet voice source. The input linguistic parameters of the FPM are: the burst cell number *bcn*, the mean silence duration *msd* and the counter state *cs*. The output linguistic param-

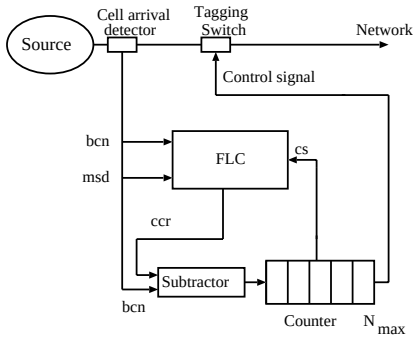

Fig. 9 FPM model.

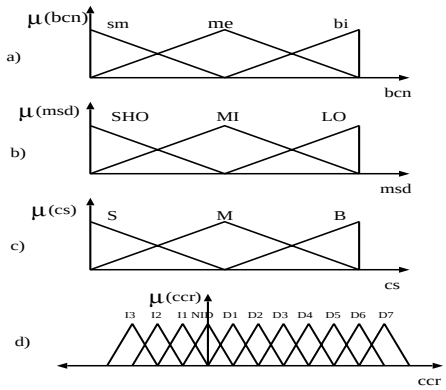

Fig. 10 Membership functions.

eter is the controlled cell rate *ccr* which enters into subtractor. The FPM model is shown in **Fig. 9**. Whereas, the membership functions for input and output linguistic parameters are shown in **Fig. 10**. The term sets of *bcn, msd, cs* are defined as:

$$T(bcn)=\{small, medium, big\}=\{sm, me, bi\};$$
$$T(msd)=\{short, middle, long\}=\{SHO, MI, LO\};$$
$$T(cs)=\{small, medium, big\}=\{S, M, B\}.$$

The set of the membership functions associated with terms in the term set of *bcn*, $T(bcn) = \{sm, me, bi\}$, are denoted by $M(bcn) = \{\mu_{sm}, \mu_{me}, \mu_{bi}\}$, where $\mu_{sm}, \mu_{me}, \mu_{bi}$ are the membership functions for *sm, me, bi,* respectively. They are given by:

$$\mu_{sm}(bcn) = f(bcn, sm_c, sm_{w0}, sm_{w1});$$
$$\mu_{me}(bcn) = f(bcn, me_c, me_{w0}, me_{w1});$$
$$\mu_{bi}(bcn) = f(bcn, bi_c, bi_{w0}, bi_{w1}).$$

The small letters *c, w0* and *w1* mean center, right width and left width, respectively.

$M(msd) = \{\mu_{SHO}, \mu_{MI}, \mu_{LO}\}$ are the membership functions for term set of *msd*. The membership functions μ_{SHO}, μ_{MI} and μ_{LO} are given by:

$$\mu_{SHO}(msd)=f(msd, SHO_c, SHO_{w0}, SHO_{w1});$$
$$\mu_{MI}(msd)=f(msd, MI_c, MI_{w0}, MI_{w1});$$
$$\mu_{LO}(msd)=f(msd, LO_c, LO_{w0}, LO_{w1}).$$

The membership functions for term set *cs* are $M(cs) = \{\mu_S, \mu_M, \mu_B\}$, and μ_S, μ_M, μ_B are given by:

$$\mu_S(cs) = f(cs, S_c, S_{w0}, S_{w1});$$
$$\mu_M(cs) = f(cs, M_c, M_{w0}, M_{w1});$$
$$\mu_B(cs) = f(cs, B_c, B_{w0}, B_{w1}).$$

We define the term set of the output linguistic parameter $T(ccr)$ as {Increase 3, Increase 2, Increase 1, Not Increase Not Decrease, Decrease 1, Decrease 2, Decrease 3, Decrease 4, Decrease 5, Decrease 6, Decrease 7}. We write for short as {I3, I2, I1, NID, D1, D2, D3, D4, D5, D6, D7}, where I2 increases more than I1 and D2 decreases more than D1 and so on.

The term set of the output membership functions, are denoted by $M(ccr)$. They are written as $\{\mu_{I3}, \mu_{I2}, \mu_{I1}, \mu_{NID}, \mu_{D1}, \mu_{D2}, \mu_{D3}, \mu_{D4}, \mu_{D5}, \mu_{D6}, \mu_{D7}\}$, and are given by:

$$\mu_{I3}(ccr) = f(ccr, I3_c, I3_{w0}, I3_{w1});$$
$$\mu_{I2}(ccr) = f(ccr, I2_c, I2_{w0}, I2_{w1});$$
$$\mu_{I1}(ccr) = f(ccr, I1_c, I1_{w0}, I1_{w1});$$
$$\mu_{NID}(ccr) = f(ccr, NID_c, NID_{w0}, NID_{w1});$$
$$\mu_{D1}(ccr) = f(ccr, D1_c, D1_{w0}, D1_{w1});$$
$$\mu_{D2}(ccr) = f(ccr, D2_c, D2_{w0}, D2_{w1});$$
$$\mu_{D3}(ccr) = f(ccr, D3_c, D3_{w0}, D3_{w1});$$
$$\mu_{D4}(ccr) = f(ccr, D4_c, D4_{w0}, D4_{w1});$$
$$\mu_{D5}(ccr) = f(ccr, D5_c, D5_{w0}, D5_{w1});$$
$$\mu_{D6}(ccr) = f(ccr, D6_c, D6_{w0}, D6_{w1});$$
$$\mu_{D7}(ccr) = f(ccr, D7_c, D7_{w0}, D7_{w1}).$$

Based on the above linguistic description of input and output parameters a FRB is constructed. The FRB forms a fuzzy set of dimensions $|T(bcn)| \times |T(msd)| \times |T(cs)|$, where $|T(x)|$ is the number of terms on $T(x)$. So, there are 27 rules in the FRB, which are shown in **Table 1**. The control rules have the following form: IF "conditions" THEN "control action". The conditions are the input linguistic parameters and the control action is the output linguistic parameter. Statements on conditions go like "the *bcn* is small" or "the *msd* is long". Likewise, statements on control action may be "increase the *ccr*" or "decrease the *ccr*".

The *ccr* depends on *bcn*, *msd* and *cs* values. The mean cell rate value is calculated as $[m = bcn/(mbd + msd)]$. To explain how we decide the *ccr* membership functions let consider rule 0. The rule 0 is as follows: if *bcn* is

Table 1 FRB.

Rule	*bcn*	*msd*	*cs*	*ccr*
0	sm	SHO	S	I3
1	sm	SHO	M	I2
2	sm	SHO	B	D2
3	sm	MI	S	D1
4	sm	MI	M	D2
5	sm	MI	B	D3
6	sm	LO	S	D1
7	sm	LO	M	D3
8	sm	LO	B	D4
9	me	SHO	S	I2
10	me	SHO	M	NID
11	me	SHO	B	D2
12	me	MI	S	NID
13	me	MI	M	D2
14	me	MI	B	D3
15	me	LO	S	D2
16	me	LO	M	D4
17	me	LO	B	D5
18	bi	SHO	S	I1
19	bi	SHO	M	D1
20	bi	SHO	B	D2
21	bi	MI	S	D2
22	bi	MI	M	D4
23	bi	MI	B	D5
24	bi	LO	S	D4
25	bi	LO	M	D6
26	bi	LO	B	D7

small, *msd* is short, and *cs* is small, then the *ccr* should be increased by I3. The mean cell rate depends on *bcn* and *msd*. If the *bcn* is small the m decreases, otherwise if *msd* is short the m increases. In the case of rule 0, the value of *cs* is small. But, the short value of *msd* may give a m value bigger than negotiated m value. Thus, the *ccr* should be increased enough in order to detect the violation of the negotiated m value. In the case of rule 1, *bcn* is the same with rule 0, but the *msd* is middle and the m value will be decreased compared with rule 0. This is why the *ccr* value should be increased less than rule 0. The other rules are built the same. It should be noted that at the same time a maximum of 6 membership functions can be activated. Therefore, some membership functions effect the strength of other membership functions.

The FPM works in the following way. The detector counts the number of cells going to the network and at the same time going to the FLC and subtractor. The parameters of the controlled source *bcn*, *msd* and the counter state parameter *cs* are the input parameters for the FLC. Based on the values of input parameters, the FLC gives an appropriate output value, which enters into subtractor. The subtractor carries out the operation $[bcn - ccr]$. If the *ccr* value is positive, the number of cells entering the counter decreases. On the other hand, if the *ccr* value is negative, the number of cells entering the counter increases. The state of the counter is expressed as $[cs = cs_0 + bcn - ccr]$, where cs_0 is the initial counter state, *bcn* is the number of cells in a burst, and *ccr* is the FLC output which indicates the number of cells that the counter state should be changed. If the source doesn't violate the negotiated parameters, the counter state is always less than the maximum counter value, so all cells are going to the network. Otherwise, if the source violates the negotiated parameters, the counter state exceeds the maximum counter value, therefore a control signal is sent to the TS to tag the violation cells. At the same time, the counter state starts from zero.

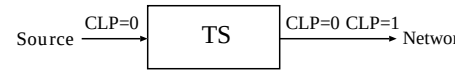

Fig. 11 TS operation scheme.

4.2 Tagging Switch Operation

The TS operation scheme is shown in **Fig. 11**. The TS is implemented via a single indicator in the ATM cell header, termed the "Cell Loss Priority" (CLP) indicator. When this indicator is set CLP=1, it signifies that the cell may be discarded in any network element along the path if the network is congested. The CLP indicator serves a dual purpose: a setting of the CLP indicator of a cell to 1 by the TS signify that the cell carries nonessential information, so this cell is discardable under congestion condition; getting of the indicator CLP=1 at the access to the network it is judged by the network that the cell is in violation of the traffic limits agreed to in the negotiated contract.

The TS can be viewed as a throughput burstiness filter which separates the source information into nonviolation traffic CLP=0 and violation traffic CLP=1. Thus, the TS reduces the impact of traffic uncertainty. By traffic policing and traffic violation tagging the total CLP=0 traffic can be handled and network utilization can be improved by CLP=1 traffic.

4.3 Description of Routing Algorithms

The network is designed to support all sources which don't violate the negotiated parameters. This means, the CLP=0 cells can be transmitted via Direct Path (DP) to the Destination Node (DN). If some sources don't use the available bandwidth, the remained bandwidth

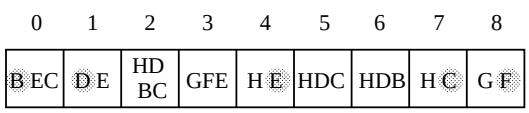

Fig. 12 A chromosome example.

can be used for transmitting the CLP=1 via DP. Otherwise, the CLP=1 cells will be transmitted to the DN via Alternate Paths (APs). The TARG algorithm is activated when a congestion situation happens in the DP to find an AP in order to save the CLP=1 cells from discarding.

The TARG algorithm is a source routing mechanism. In the source routing mechanism, a complete path from Source Node (SN) to DN is decided from SN. The SN knows all paths and the complete information about the network. Therefore, it is possible to select a path efficiently. Also, if the data is large in quantity, it is possible to divide the flow data by using different paths. Furthermore, the source routing mechanism has a fast decision because there are not computations at intermediate nodes.

The TARG algorithm is based on GAs. The most important factor to achieve efficient genetic operations is gene coding. In the GLBR algorithm, the genes are put in a chromosome in the same order the nodes are in a path, so the chromosomes have different sizes which result in complex genetic operations. In order to simplify the genetic operations of GLBR, in the TARG algorithm, the network is expressed by a tree network and the genes represent the tree junctions. A chromosome example is shown in **Fig. 12**. The genes in a chromosome have two states "active" and "inactive". A gene is called "active" if the junction is in the path, otherwise the gene is in "inactive" state. The genetic operations are carried out in the "active" genes. Each gene includes information of the adjacent nodes. All chromosomes have the same length, which results in easy genetic operations.

The genetic operation procedure for both algorithms is as follows.

```
void GeneticOperation()
{
   while(pop_size < POPULATION_SIZE) {
      Selection();
      Crossover();
   }
   Mutation();
   Elitist();
}
```

In the selection operation, first, an initial population is selected. In the selected population, the ranking selection model is used to select two individuals in order to carry out the genetic operation. The ranking model ranks each individual by their fitness. The rank is decided based on the fitness and the probability is decided based on the rank. The individual fitness is based on delay time. If the delay time is small, the individual fitness is high. When the rank is high, the probability of individuals is high.

The crossover operation used is the single point crossover. The crossover operation for the GLBR algorithm is shown in following.

```
void Crossover()
{
   if(within crossover rate) {
      crossover point decision;
      crossover operation;
      CheckNode();
   }
}
```

In the GLBR, after the crossover operation is finished, the algorithm should check whether a node is passed many times or not. If a node is passed many times the route will be long, so the chromosome will be also long and the algorithm complexity is increased.

The crossover operation for TARG algorithm is shown in following.

```
void Crossover()
{
   if(within crossover rate) {
      crossover point decision;
      crossover operation;
   }
}
```

By using the tree model, the TARG algorithm can avoid the routing loops, which results in the response time improvement.

In the mutation operation, the genes are chosen randomly in the range from zero up to mutation probability $p\_mutation \leq \frac{1}{\ell}$, where l is the chromosome length.

The mutation operation for the GLBR algorithm is shown in following.

```
void Mutation()
{
   if(within mutation rate) {
      decision of the population;
      decision of the locus;
      mutation operation;
      CheckRoute();
```

```
    }
}
```

In the GLBR, the genes are put in a chromosome the same way as they are in a route. After the mutation operation, the route between a object node and the selected node may not exist. Therefore, the algorithm needs to check out whether a route exist or not. For example in Fig. 4, let consider that the node E is changed to G by mutation. The route between A and H should be "A, G, H", but this route doesn't exist.

The mutation operation for the TARG algorithm is shown in following.

```
void Mutation()
{
   if(within mutation rate) {
      decision of the population;
      decision of the locus;
      mutation operation;
   }
}
```

In the TARG algorithm, by using the tree model the selected route always exists, so the algorithm doesn't need to check the validity of the selected route.

After the crossover and mutation, the elitist model is used. Based on the elitist model the individual which has the highest fitness value in a population is left intact in the next generation. Therefore, the best value is always kept and the routing algorithm can converge very fast to the desired delay time.

5. Simulation Results

We make the following assumptions for simulations: the source is directly connected to the ATM network; the source peak cell rate is controlled separately by a PM; the delay time is set at each link; a sudden congestion situation is assumed in the DP.

We generate the burst and the silence period in an independent way. The distribution functions are exponential and the density functions f(.) are expressed as: $f_B(b) = 1/mbd * \exp^{-mbd*b}$ and $f_S(s) = 1/msd * \exp^{-msd*s}$, for the burst and silence, respectively. We use the packet voice source for simulations, because this source is a prototype of an off-on source and is considered as the worst case traffic pattern [1].

The packed voice parameters are as follows:

pcr0 $= 32\,\text{kb/s} \approx 62\,\text{cell/s}$,

mbd0 $= 352\,\text{ms}$, msd0 $= 650\,\text{ms}$,

m $= 11.2\,\text{kb/s} \approx 22\,\text{cell/s}$.

Table 2 Assignment of values for input and output linguistic parameters.

	bcn	
$sm_c = 0$	$sm_{w0} = 0$	$sm_{w1} = 35$
$me_c = 35$	$me_{w0} = 35$	$me_{w1} = 35$
$bi_c = 70$	$bi_{w0} = 35$	$bi_{w1} = 0$
	msd	
$SHO_c = 0$	$SHO_{w0} = 0$	$SHO_{w1} = 1000$
$MI_c = 1000$	$MI_{w0} = 1000$	$MI_{w1} = 1000$
$LO_c = 2000$	$LO_{w0} = 1000$	$LO_{w1} = 0$
	cs	
$S_c = 0$	$S_{w0} = 0$	$S_{w1} = 15$
$M_c = 15$	$M_{w0} = 15$	$M_{w1} = 15$
$B_c = 30$	$B_{w0} = 15$	$B_{w1} = 0$
	ccr	
$I3_c = -30$	$I3_{w0} = 10$	$I3_{w1} = 10$
$I2_c = -20$	$I2_{w0} = 10$	$I2_{w1} = 10$
$I1_c = -10$	$I1_{w0} = 10$	$I1_{w1} = 10$
$NID_c = 0$	$NID_{w0} = 10$	$NID_{w1} = 10$
$D1_c = 10$	$D1_{w0} = 10$	$D1_{w1} = 10$
$D2_c = 20$	$D2_{w0} = 10$	$D2_{w1} = 10$
$D3_c = 30$	$D3_{w0} = 10$	$D3_{w1} = 10$
$D4_c = 40$	$D4_{w0} = 10$	$D4_{w1} = 10$
$D5_c = 50$	$D5_{w0} = 10$	$D5_{w1} = 10$
$D6_c = 60$	$D6_{w0} = 10$	$D6_{w1} = 10$
$D7_c = 70$	$D7_{w0} = 10$	$D7_{w1} = 10$

5.1 Policing Mechanisms Simulation Results

The PM selectivity is measured as the violation probability V_p, that the PM will detect a cell as excessive which is considered as a violating cell. The ideal behavior will be for V_p to be zero when the mean cell rate is up to nominal one, and $V_p = (\gamma - 1)/\gamma$ for $\gamma > 1$, where γ is the long term actual mean cell rate of the source normalized to the negotiated mean cell rate.

The values for input and output linguistic parameters are assigned as shown in **Table 2**. The assignment is based on the range of linguistic parameters and the number of the membership functions. The maximum burst duration and the maximum silence duration are chosen 1 second and 2 seconds, respectively. The counter size is chosen 30 cells.

We carried out many simulations to evaluate the behavior of the FPM, but for the sake of space we show in **Fig. 13** only the selectivity characteristic of the FPM. Some studies proposed so far [1),2)] show that the LBM has a better performance compared with the other conventional PMs. However, in Ref. 4) is shown that the LBM has performance limitation for user parameter control in ATM networks. To deal with these limitations, in Refs. 6) and 7), a fuzzy policer is proposed to control the mean

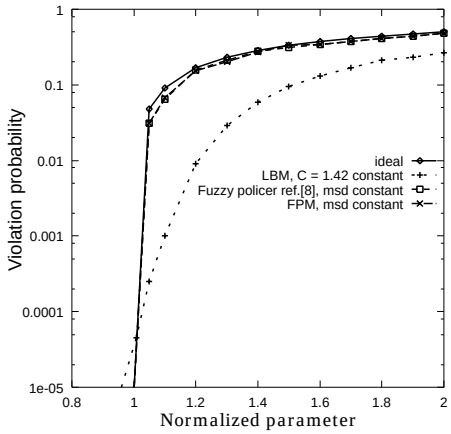

Fig. 13 Performance comparison of PMs.

cell rate of the bursty sources. The difference between the fuzzy policer and our FPM is that the fuzzy policer is a window-based PM, while FPM is a leaky-bucket-based PM. The performance of PMs is shown in Fig. 13. All PMs are policing the mean cell rate of the packet voice source. The policed mean cell rate of the LBM is $C \cdot mcr$, where C is the over dimensioning factor of the LBM. The performance characteristics of fuzzy policer and FPM are very closed to the ideal characteristic compared with the LBM. The comparison between our proposed FPM and the LBM shows that the FPM has a better selectivity characteristic than the LBM. For instance, with normalized parameter about 1.2, there is an improvement of violation probability of over one order of magnitude as compared with LBM. The FPM starts to tag (discard) the cells when the mean cell rate is more than 22 cell/s, while the LBM starts to discard the cells before the mean cell rate is 22 cell/s. This show that our FPM have a good responsiveness to parameter violation compared with LBM. In order to achieve a better selectivity characteristic, the LBM should have a high value for the counter limit which means a poor dynamic response, or a high value for C parameter which reduces the capacity to detect the violation cells. Thus, our FPM improves both the selectivity and responsiveness characteristics. Furthermore, our FPM can be used to police a set of sources in time sharing. In the case of packet voice source, it can police almost a thousand of sources (see Section 6.1).

5.2 Routing Algorithms Simulation Results

In Ref. 12) is shown a performance comparison between the GLBR algorithm and conventional SPF and RIP algorithms. The GLBR algorithm has a better behavior compared with SPF and RIP algorithms. Therefore, in following, we compare the performance of TARG algorithm with GLBR algorithm.

Table 3 Simulation parameters.

population size	5, 10, 20, 30
crossover rate (%)	70, 80, 90, 100
mutation rate (%)	1, 5, 10, 20

To compare both algorithms, the first population is selected the same. After the congestion situation happens in the DP, the TARG and GLBR algorithms search for a new path in order to avoid the congested path. The genetic operations are repeated until the path with smallest delay is found or the initialized generation size is achieved. The parameters used in simulations are shown in **Table 3**. Based on the data obtained from the TARG and GLBR algorithms, the characteristics of delay time versus simulation step are depicted.

The TARG and GLBR algorithms are compared for different population sizes, crossover rates and mutation rates, but for the sake of space, we have shown in **Table 4** only the simulation data for population sizes 10 and 20. The mutation rate changes from 1% to 20% and the crossover rate changes from 70% to 100%. The values inside the table show the search rate when both algorithms find the shortest path. In all simulations, the TARG algorithm finds the shortest path faster than the GLBR algorithm.

During the simulations, for the population size 5 the search rate was high. This means, the number of genetic operations to find the shortest path increases. For the population size 10, the result was improved, and when the population size was 20 the result was improved much more. However, when the population size was 30, both algorithms could not achieve an efficient search, because the genetic operations become very complex. We conclude that, the best population size is 20. The decision of the best population size is a trade-off between diverse constrains. If the population size is small, the algorithms converge fast to a local minima, but they may not give the best response. Otherwise, if the population size is big, the algorithms need time to carry out the genetic operations. The change of crossover rate doesn't have too much effect on the results of algorithms. On the other hand, the change of mutation rate has a

Table 4 Search rate (%) of the TARG and GLBR algorithms.

population size 10								
mutation	crossover rate 70%		crossover rate 80%		crossover rate 90%		crossover rate 100%	
rate	TARG	GLBR	TARG	GLBR	TARG	GLBR	TARG	GLBR
1%	49.5	52.0	46.2	54.2	49.5	56.4	44.6	47.6
5%	36.3	44.0	40.9	43.6	40.8	46.6	39.3	41.1
10%	27.7	37.1	31.7	41.1	30.9	33.9	31.5	39.1
20%	30.5	35.9	27.3	33.9	19.9	32.9	20.4	28.1
population size 20								
mutation	crossover rate 70%		crossover rate 80%		crossover rate 90%		crossover rate 100%	
rate	TARG	GLBR	TARG	GLBR	TARG	GLBR	TARG	GLBR
1%	17.4	21.1	14.9	21.3	18.6	21.3	17.4	20.9
5%	13.2	16.8	16.6	23.7	14.7	18.7	17.8	18.4
10%	16.9	18.7	16.1	20.9	17.9	26.7	15.6	25.7
20%	12.9	15.9	13.4	21.2	12.6	22.3	15.9	26.7

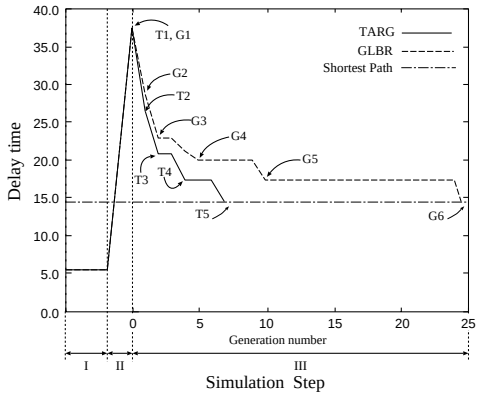

Fig. 14 Performance comparison of TARG and GLBR algorithms.

great effect in the algorithms performance. If the mutation rate is small, the created population types are limited. Otherwise, if the mutation rate is big, the delay time doesn't decrease. Therefore, the algorithms need time to find the shortest path. We conclude that a mutation rate about 10% is a good mutation rate.

Figure 14 shows the characteristics of delay time versus simulation step for the TARG and GLBR algorithms. The simulation step consists of three parts: step I is the communication state, step II is the congestion state and step III is the algorithm operation state. Step III (generation size) shows how many genetic operations are needed in order to find the shortest path. The TARG algorithm can find the shortest path faster than the GLBR algorithm. During a lot of simulations, we surveyed that, the time for generating a new population with the TARG algorithm is about 6 times faster than with the GLBR algorithm. The GLBR genetic operations complexity is because new individuals (paths) may not exist, so the GLBR algorithm should generate new populations to get the shortest path. The individuals of the GLBR algorithm are created partially. Therefore, the individuals always will be generated in the same part of the network. On the other hand, the TARG algorithm can create different types of individuals, which result in a fast evolution of the algorithm.

Table 5 shows a comparison between two algorithms for different Generation Number (GN). The labels inside the table are obtained in the points where the delay time has changed. The labels T2–T5 are for TARG algorithm and labels G2–G6 are for GLBR algorithm. In labels T1 and G1 the GN is zero. These points are the points where both algorithms start the search. The search results are shown up to the rank number 7. The TARG algorithm has achieved the rank number 7 after 7 generations. The selected route is "ABDCEHLNST". While in this stage, the GLBR algorithm is in the route "ABDHLNMPQRST" and the rank number is 36. The GLBR algorithm needs 24 generations for the rank number 7.

For further evaluation of TARG and GLBR algorithms, we use the following parameters: the mean number of genetic operations to find the shortest route G_a; the ratio (%) of [the number of genetic operations when the algorithm didn't find a route] to [the number of simulations] N_r; the ratio (%) of [the number of populations used during the search] to [the complete number of populations] P_r; the ratio (%) of [the number of routes used during the search] to [the complete number of routes] T_r; the mean rank number of the shortest route R_a.

Considering the following terms: the number of populations used during the search *nr_populations_ds*, the complete number of populations *all_populations_number*, the number of routes used during the search

Table 5 Comparison for different GN.

GN	TARG				GLBR			
	Label	Route	Delay	Rank	Label	Route	Delay	Rank
1	T2	ABDCEHLNMPOQRST	26.5	175	G2	ABDCEHIJLNMPOQRST	28.9	203
2	T3	ABDHLKMPOQRST	20.9	54				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4	T4	ABDHLKMPQRST	17.3	19	G3	ABDHEHLNMPQRST	21.2	69
5	⋮	⋮	⋮	⋮	G4	ABDHLNMPQRST	20.0	36
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	T5	ABDCEHLNST	14.7	7	⋮	⋮	⋮	⋮
⋮					⋮	⋮	⋮	⋮
10					G5	ABDHLKMPQRST	17.3	19
⋮					⋮	⋮	⋮	⋮
24					G6	ABDCEHLNST	14.7	7

Table 6 Efficiency parameters.

	PS 10		PS 20	
	TARG	GLBR	TARG	GLBR
G_a	7.4	12.4	1.8	7.5
N_r	0.0	0.0	0.0	0.0
P_r	10.2	17.1	4.9	20.7
T_r	3.2	3.0	3.8	3.4
R_a	5.4	6.1	4.4	4.7

$nr\_searched$, the complete number of routes all_route_number, the number of simulation sn, the number of genetic operations when the algorithm didn't find a route not_found_rn, the number of genetic operations in the i-th simulation g_i, and the rank number of the i-th simulation r_i, the parameters G_a, N_r, P_r, T_r, and R_a can be calculated as follows:

$$G_a = \frac{\sum_{i=1}^{sn} g_i}{sn};$$

$$N_r = \frac{not\_found\_rn}{sn} \times 100\ (\%);$$

$$P_r = \frac{nr\_populations\_ds}{all\_populations\_number} \times 100\ (\%);$$

$$T_r = \frac{nr\_searched}{all\_route\_number} \times 100\ (\%);$$

$$R_a = \frac{\sum_{i=1}^{sn} r_i}{sn}.$$

The efficiency parameters are shown in **Table 6**. In all simulations, the TARG algorithm has better efficiency parameters and small number of genetic operations compared with the GLBR algorithm.

The TARG algorithm can cope with small scale networks (tens of nodes). In order to cope with more large scale networks, we propose a new algorithm to reduce the search space of the TARG algorithm. We call this algorithm Search Space Reduction Algorithm (SSRA) and its flowchart is shown in **Fig. 15**. The key element of SSRA is Effective Topology (ET) extraction. The ET extraction of a network is defined as the topology based on which a path is constructed for a connection. In order to extract the ET, the network connectivity information, link and node metrics, and QoS requirement of the new connection are required. We use the Equivalent capacity (Ec) to specify the QoS demand of a new connection. In order to have a low overhead processing time, we consider the Available Bandwidth (AB) as the only link and node metrics. If a Link Available Bandwidth (LAB) or Node Available Bandwidth (NAB) is less than Ec of a connection, this means that every path which passes via this link or node cannot satisfy the connection requirements.

First, the SSRA based on the required Ec checks all links in the network whether their AB satisfies or not the Ec. If a LAB doesn't satisfy the Ec then the link is excluded from ET. Otherwise, the link is included in the ET and the next link is checked. The procedure is repeated until all links are finished. Next, the SSRA checks all nodes in the network, whether their AB satisfies the Ec or not. If the NAB doesn't satisfy the Ec then the node is excluded from the ET. Otherwise, the node is included in the ET and the next node is checked. The procedure is repeated until all nodes are finished. Finally, after all links and nodes are checked, the network ET is constructed and the com-

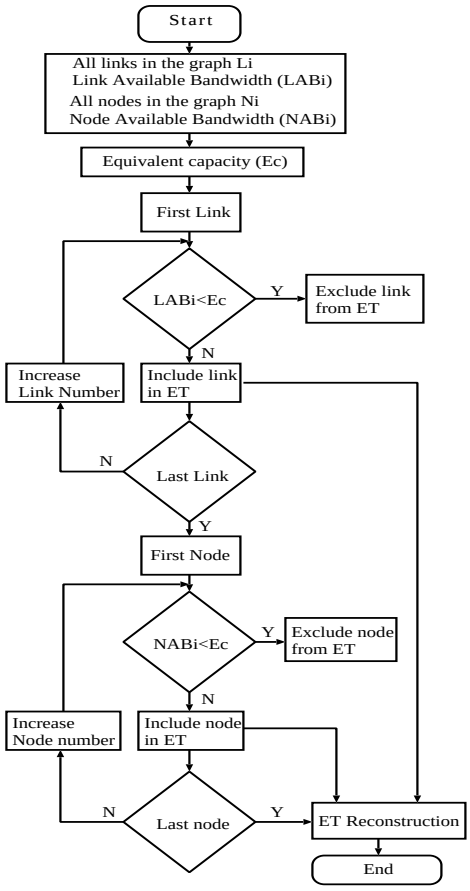

Fig. 15 SSRA flowchart.

plete procedure is finished.

By using the SSRA, a network with many nodes and links will be reduced in a network with a small number of nodes and links. Thus, the TARG will be able to cope with more large scale networks (hundred of nodes).

6. Implementation Issues

6.1 FPM Implementation Issues

Many ultra-low cost fuzzy chips exist and are recently proposed. We consider as a fuzzy chip for implementation, the parallel architecture proposed in Ref. 16). This processor has the following characteristics. The hardware implementation of the processor comprises 4 Fuzzy Processing Units (FPUs). The clock frequency at which each processor operates is 60 MHz. The speed of this fuzzy chip is about 77000 Fuzzy Logic Inference Operations Per Second (FLIOPS), if one FPU is operating.

The processing speed depends on the statistical characteristics of the source to be controlled. Considering a packetized voice source with a peak cell rate in a burst of 32 kb/s and an ATM cell size of 53 bytes, the inter arrival cell time t_c is 12 ms. This is the maximum time limit by which the FPM has to infer the output. The FPM begins to process the rules when the input parameters are all available. If we denote the processing time of the FPM with t_f, the value of t_f should be smaller than t_c in order to prevent a cell arriving at the beginning of the new cycle to escape the control action. Let express the time constraint in terms of FLIOPS. The FPM has to give a minimum performance of $1/t_c$. The t_c is 12 ms, so the performance required is about 84 FLIOPS. The fuzzy processor has a speed 77000 FLIOPS. Thus, the FPM is capable of policing about thousand sources. This results in an improvement of the exploitation of hardware when the fuzzy processor is used to police a set of sources in time sharing.

6.2 TARG Implementation Issues

The implementation of methods based on GAs have been a burden for the GAs application. Also, the production of parallel processing architectures to process GAs operations are slow compared with fuzzy chips. However, now, there are many attempts to implement GAs in parallel processing architectures [17),18)].

The TARG algorithm is able to cope with small and medium scale (hundred of nodes) ATM networks. But, when the number of nodes increases the number of tree junctions increases too. Thus, the chromosome length will be longer and the execution of genetic operations will take more time. To deal will this problem, the authors are considering two approaches: 1) the implementation of TARG algorithm in a parallel processing architectures [18)]; and 2) building a new distributed routing architecture based on cooperative agents.

7. Conclusions

In this paper, we proposed an intelligent PRM for ATM networks based on FL and GAs. After giving a brief introduction of the FL and GAs, we treated the source and network models. Next, we described the system model and its elements. In the following, we discussed the simulation results. Finally, we treated some implementation issues. The behavior of FPM and TARG algorithm was investigated by simulations. From the simulation results, we conclude:

- the FPM efficiently monitors the mean cell rate of the packet voice source;

- the FPM has a good responsiveness characteristic;
- the FPM selectivity characteristic approaches very close to the ideal characteristic required for a PM;
- the performance of the FPM is better than the LBM;
- the TARG algorithm has an efficient search;
- the TARG algorithm has a faster evolution compared with the GLBR algorithm;
- the TARG algorithm can avoid the routing loops, so the algorithm doesn't lose time searching in the routing loops;
- in the TARG algorithm, the searched route always exists, therefore the algorithm doesn't need to check the route validity as in GLBR algorithm;
- the GLBR genetic operations are more complex than the TARG genetic operations.

The performance evaluation via simulations shows that the proposed PRM has a better behavior compared with traditional PMs and RAs. Furthermore, by using the tagging function of the PM and handling CLP=1 traffic the network utilization can be improved.

Additional work is in progress to provide detailed quantitative evidence for the global performance of the proposed PRM.

The authors are working now to build a new architecture based on cooperative agents in order to cope with very large scale networks. The simulation results of the proposed architecture will be reported in the future.

References

1) Rathgeb, E.: Modeling and Performance Comparison of Policing Mechanisms for ATM Networks, *IEEE J. Select. Areas Commun.*, Vol.SAC-9, No.3, pp.325–334 (1991).
2) Buttó, M., Cavallero, E. and Tonietti, A.: Effectiveness of the "Leaky Bucket" Policing Mechanism in ATM Networks, *IEEE J. Select. Areas Commun.*, Vol.SAC-9, No.3, pp.335–342 (1991).
3) Guillemin, F., Boyer, P., Dupis, A. and Romoeuf, L.: Peak Rate Enforcement in ATM Networks, *Proc. IEEE Infocom'92*, pp.6A.1.1–6A.1.6 (1992).
4) Yamanaka, N., Sato, Y. and Sato, K.: Performance Limitation of the Leaky Bucket Algorithm for ATM Networks, *IEEE Trans. Comm.*, Vol.43, No.8, pp.2298–2300 (1995).
5) Baransel, C., Dobosiewicz, W. and Gburzynski, P.: Routing in Multihop Packet Switching Networks: Gb/s Challenge, *IEEE Network*, Vol.9, No.3, pp.38–60 (1995).
6) Catania, V., Ficili, G., Palazzo, S. and Panno, D.: A Comparative Analysis of Fuzzy Versus Conventional Policing Mechanisms for ATM Networks, *IEEE/ACM Trans. Networking*, Vol.4, No.3, pp.449–459 (1996).
7) Catania, V., Ficili, G., Palazzo, S. and Panno, D.: Using Fuzzy Logic in ATM Source Traffic Control: Lessons and Perspectives, *IEEE Commun. Magazine*, Vol.34, No.11, pp.70–81 (1997).
8) Bonde, A.R. and Ghosh, S.: A Comparative Study of Fuzzy Versus "Fixed" Thresholds for Robust Queue Management in Cell-Switching Networks, *IEEE/ACM Trans. Networking*, Vol.2, No.4, pp.337–344 (1994).
9) Cheng, R. and Chang, C: Design of a Fuzzy Traffic Controller for ATM Networks, *IEEE/ACM Trans. Networking*, Vol.4, No.3, pp.460–469 (1996).
10) Khalfet, J. and Chemouil, P.: Application of Fuzzy Control to Adaptive Traffic Routing in Telephone Networks, *Information and Decision Tech.*, Vol.19, No.4, pp.339–348 (1994).
11) Habib, I. (Ed.): *Neurocomputing in High-Speed Networks*, Special Issue of *IEEE Commun. Magazine*, Vol.33, No.10 (1995).
12) Munetomo, M., Takai, Y. and Sato, Y.: An Adaptive Routing Algorithm with Load Balancing by a Genetic Algorithm, *Trans. Information Processing Society of Japan*, Vol.39, No.2, pp.219–227 (1998).
13) Zadeh, L.A.: Fuzzy Logic, Neural Networks, and Soft Computing, *Commun. ACM*, Vol.37, No.3, pp.77–84 (1994).
14) Dubois, D., Prade, H. and Yager, R. (Eds.): *Fuzzy Sets for Intelligent Systems*, Morgan Kaufman (1993).
15) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).
16) Catania, V. and Ascia, G.: A VLSI Parallel Architecture For Fuzzy Expert Systems, *International J. Pattern Recognition and Artificial Intelligence*, Vol.9, No.2, pp.421–447 (1995).
17) Cantu-Paz, E.: Designing Efficient and Accurate Parallel Genetic Algorithms, Ph.D. Thesis, University of Illinois at Urbana-Champaign (1999).
18) Tang, K.S., Man, K.F. and Kwong, S.: Parallel Genetic Algorithms: Implementable Hardware Solutions, *IEEE Circuits and Systems Society Newsletter*, Vol.10, No.2, pp.3–11 (1999).

(Received January 20, 2000)
(Accepted September 7, 2000)

Leonard Barolli was born in Bilisht, Albania. He received the B.E. and Ph.D. degrees in 1989 and 1997 from Tirana University and Yamagata University, respectively. From April 1997 until March 1999, he was working as a Post Doctor Fellow Researcher of JSPS at Faculty of Engineering, Yamagata University. From April 1999, he is working as a Research Associate at Department of Public Policy and Social Studies, Yamagata University. His research interests include traffic control in ATM networks, fuzzy control, genetic algorithms and agent-based systems. He is a member of SOFT and IPSJ.

Akio Koyama was born in Yonezawa, Yamagata prefecture. He received the B.E. and Ph.D. degrees in Information Engineering from Yamagata University in 1987 and 1998, respectively. From April 1981 until March 1999, he was working as a technical staff at Faculty of Engineering, Yamagata University. From April 1999, he is working as an Assistant Professor at Department of Computer Software, the University of Aizu. His current research interests include distributed systems, parallel computer architecture, LAN protocols, traffic and congestion control in ATM networks. He is a member of IEEE Computer Society, IPSJ and IEICE.

Takako Yamada was born in Osaka. She received the B.Ec. in 1983 from Tohoku University and M.S. and Ph.D. degrees from Tokyo Institute of Technology in 1986 and 1995, respectively. From April 1995, she is working as an Associate Professor at Department of Public Policy and Social Studies, Yamagata University. Her research interests include modeling and performance evaluation, telecommunication networks, and mobile communication. She is a member of Operational Research Association of Japan, IPSJ and IEICE.

Shoichi Yokoyama was born in Marugame, Kagawa prefecture. He received the B.E. and Ph.D. degrees from University of Tokyo in 1972 and 1987, respectively. In 1972 he was with the Electrotechnical Laboratory, Information Science Division. Since 1993 he has been a Professor at Faculty of Engineering, Yamagata University. His current research interests include natural language processing, electronic dictionary, machine translation, and artificial intelligence. He is a member of IPSJ, SICE, JASJ, ACL and Euralex.