

# 공장자동화용 네트워크를 위한 통합성능관리기의 개발

## Development of Integrated Network Performance Manager for Factory Automation Networks

이 상 호, 김 인 준, 이 경 창, 이 석  
(Sang-Ho Lee, In-Joon Kim, Kyung-Chang Lee, and Suk Lee)

**Abstract** : This paper focuses on development of a performance manager for IEEE 802.4 token bus networks to serve large-scale integrated systems. In order to construct the management algorithm, the principles of fuzzy logic, genetic algorithm, and neural network have been combined to represent human knowledge and to imitate of human inference mechanism. Through the simulation experiments, it is shown that the proposed performance manager is capable of improving the network performance without a priori knowledge.

**Keywords** : token bus protocol, priority mechnism, fuzzy network performance manager, data latency, self-organization

### I. 서론

컴퓨터 통합생산을 위한 컴퓨터 네트워크의 생산시스템 적용 예가 그림 1에 나타나있는데, 이러한 네트워크에 의해 교환되는 정보들은 다양한 종류의 데이터들로 표현되고 있다. 작업장 레벨에서는 센서 신호나 제어 신호, 고장 진단 및 유지/관리 데이터 등의 통신이 이루어지며, 짧은 메시지 특성과 함께 주기적이면서 빈번히 발생하는 편이다. 또한, 전송지연(data latency, 큐에서의 메시지 대기시간과 전송시간의 합)에 대한 제약도 상당히 큰 편이라고 할 수 있다. 이에 반해, 생산시스템의 상위 레벨에서는 CAD 파일이나 공정 총괄과 같은 문서 파일들이 비교적 긴 형태로 비주기적인 특성을 가지고 가끔씩 전송되며, 전송지연에 대한 제약도 그다지 크지 않은 편이다. 따라서, 대규모 공장과 같은 곳에서, 네트워크에 의해 통합된 생산시스템을 구축하기 위해서는 시간에 대해 긴급성을 요하는 메시지들을 전송지연과 같은 요구 조건에 만족되도록 설계 및 관리할 필요가 있다 [1][2][3][4].

공장자동화용 네트워크의 표준 통신 프로토콜인 MAP (Manufacturing Automation Protocol)[5][6]에서는 물리 계층의 표준으로 IEEE 802.4 토큰버스 프로토콜을 선정하였으며, 여기에는 메시지 전송을 위해 네 가지의 우선순위를 가진다 이러한 우선순위도구(priority mechanism)에는 네 가지의 카운트-다운(count-down) 타이머, 즉 THT(Token Holding Timer)와 TRTi(Token Rotation Timer  $i, i = 4, 2, 0$ )와 각각의 큐(queue)를 가지는데, 이러한 우선순위도구의 목적은 발생된 메시지를 종류에 따라 네 가지 우선순위로 나누어 각기 다른 큐에 저장하여, 우선순위가 높은 메시지를 우선적으로 전송하도록 하기 위한 것이다. 이것은 네트워크의 통신 부하와 통신 자원

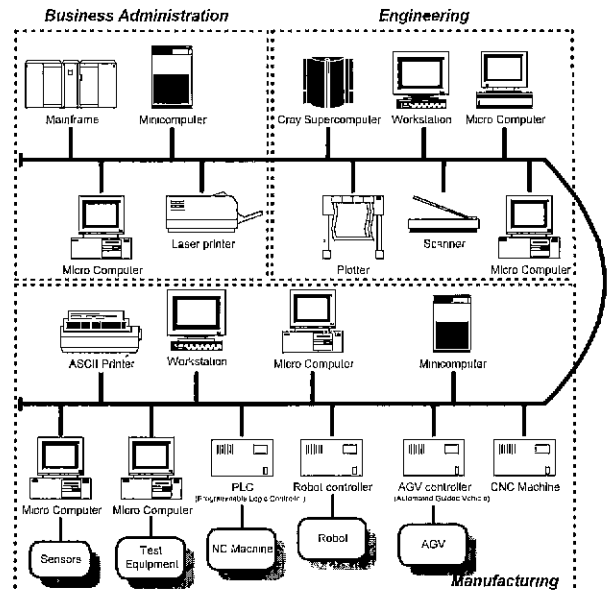


그림 1. 컴퓨터 통합생산을 위한 컴퓨터 네트워크의 적용 예.

Fig. 1. Schematic diagram of computer integrated manufacturing architecture.

이 시간에 따라 변화하는 특성을 가지기 때문으로 프로토콜 변수인 타이머들에 할당되는 값에 따라 네트워크의 전송지연, throughput 등과 같은 성능 특성이 달라진다. 따라서, 프로토콜 변수는 네트워크의 동적인 환경에 따라 자동적으로 조정되어야 할 필요가 있다

이를 위해 퍼지 논리(fuzzy logic)를 사용하였는데, 성능관리 전문가가 갖고 있는 네트워크의 성능과 통신규약 변수의 관계에 대한 지식과 새로운 변수값의 결정에 관한 지식을 내포하도록 제어규칙을 설정하였다. 이상과 같은 퍼지 기법을 도입하여 각 우선순위의 타이머와 큐 용량을 복합적으로 다루는 퍼지 네트워크 성능관리기(Fuzzy Network Performance Manager, FNPM)를 개발[7][8][9]하였으며, 시뮬레이션 언어인 SIMAN[10][11]을 이용

접수일자 : 1998. 4. 20, 수정완료 : 1999. 4. 20

이상호 · 국방과학연구소

김인준, 이경창 : 부산대학교 생산기계공학과

이 석 : 부산대학교 기계공학과

해 개발된 IEEE 802.4 토큰버스 네트워크 시뮬레이션 모델을 통해 그 효용성을 입증하였다. 그러나, 시간에 따라서 통신부하가 달라지는 네트워크의 상황을 고려하여 인간의 지식과 경험을 FNPM의 지식베이스로 변환시킬 수 있는 체계적인 방법이 필요하며, 소속함수를 스스로 조절할 수 있는 기능도 요구된다.

따라서, 유전자 알고리즘(genetic algorithm)[12][13]을 이용하여 GA 최적화기(optimizer)를 구성함으로써, FNPM의 지식베이스를 자기구성한다. 하지만, GA의 이러한 적용은 목적함수(objective function)를 필요로 하는데, 컴퓨터 네트워크에 있어서 주어진 스테이션(station)의 개수나 트래픽(traffic)과 같은 조건하에서 전송지연을 계산할 수 있는 해석적인(analytical) 방법이 존재하지 않는다. 따라서, 시뮬레이션을 통해 얻는 것이 일반적이라 할 수 있으며, 컴퓨터 시뮬레이션을 통해 유전자 알고리즘의 모든 개체에 대한 적합도(fitness)를 계산하게 되면, 관리기의 계산 부담이 매우 클 뿐만 아니라, 시뮬레이션에 소요되는 시간으로 인해 실시간 적용이 어려워져 개체군의 크기에 제한이 따르게 된다

이를 극복하기 위해, 본 연구에서는 신경망(neural networks)이 갖는 학습에 의한 적응성과 병렬 분산 처리에 의한 빠른 계산 능력 등의 장점을 이용하여[14][15][16], IEEE 802.4 토큰버스 네트워크의 특성을 신경망에 모사시킨 모델(Neural Network Approximation Model, NNAM)을 개발한다 즉, FNPM은 실시간으로 네트워크의 성능관리를 수행하고, 여기서 얻어진 네트워크 특성은 신경망의 학습 자료가 된다. 학습된 신경망은 유전자 알고리즘이 최적의 지식베이스를 탐색할 수 있는 환경으로 제시되며, 모든 개체의 적합도를 빠른 시간내에 평가할 수 있어 많은 시간의 감소와 함께 지식베이스의 계속된 향상으로 보다 효율적인 성능관리를 수행할 수 있도록 한다. 이상과 같이, 본 논문에서는 퍼지논리와 유전자 알고리즘, 신경망 등을 복합적으로 사용한 통합 네트워크 성능관리기(Integrated Network Performance Manager, INPM)를 개발하였다.

II. 퍼지 네트워크 성능관리기(FNPM)

통신부하와 네트워크의 성능은 밀접한 관계가 있는데, 주어진 통신부하에 대해서 네트워크의 성능과 프로토콜 변수 사이의 관계를 바탕으로 성능을 향상시키기 위한 프로토콜 변수들의 체계적인 조절 방법이 없는 것이 사실이다. 따라서, 종래에는 네트워크 관리자의 경험을 바탕으로 하여 수동적으로 조절되었기 때문에 바람직한 결과를 얻지 못하는 경우가 많았다. 따라서, 이와 같은 문제점을 극복하고 보다 효율적인 네트워크의 관리기능을 수행하기 위하여, IEEE 802.4 토큰버스 네트워크에 퍼지 기법을 도입한 FNPM을 개발하였다[7][8]. 개발된 FNPM은 비교적 단순한 구조와 함께, 성능과 관계된 변수를 상황에 따라서 자동적으로 조절시키는 관리기능을 갖는다. 이와 같은 FNPM의 목적은 한정된 네트워크 용량에 있어 각 우선순위의 타이머들을 적절히 할당함으로써 네

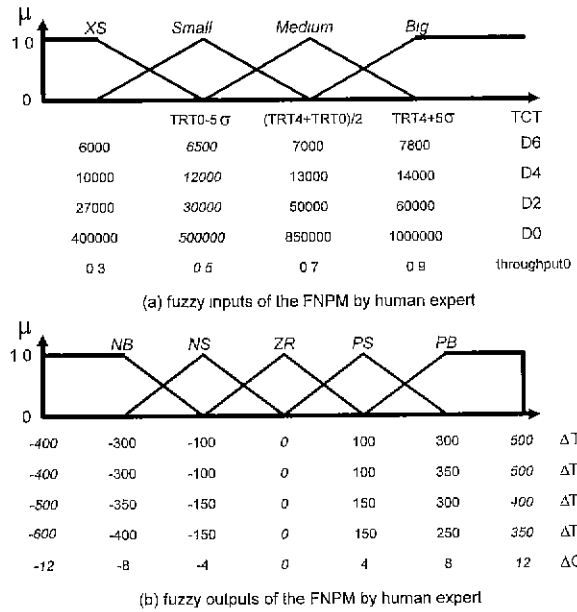


그림 2. 전문가에 의한 소속함수. Fig. 2. Membership functions by human expert.

트위크의 전송지연을 감소시켜 성능을 향상시키는 것이다. FNPM은 각 우선순위의 통신가능시간이라고 할 수 있는 타이머 할당과 메시지의 버퍼라고 할 수 있는 큐 용량을 복합적으로 조절하도록 하였다. 먼저, 입력 변수로는 그림 2(a)와 같이 토큰이 논리적 링을 한바퀴 도는데 걸리는 시간인 토큰순환시간(Token Circulation Time, TCT)과 그것의 표준편차가 필요하다. 그리고, 4가지 우선순위의 전송지연(D6, D4, D2, D0), 우선순위0의 throughput이 퍼지 입력의 형태로 받아들여지고, TRT0의 값은 보통(crisp)의 값으로 고려되었다. 특히, 전송지연에 대해서 Small의 소속함수 값이 1에 해당하는 전송지연은 네트워크 관리자가 해당 우선순위의 메시지에 대한 최대 허용 전송지연 시간으로 정의하였다. 또한, 출력변수로는 그림 2(b)와 같이 각 우선순위의 통신 가능 시간을 결정짓는 각 우선순위의 타이머 변화량( $\Delta$ THT,  $\Delta$ TRT4,  $\Delta$ TRT2,  $\Delta$ TRT0)과 우선순위0의 큐 용량의 변화( $\Delta$ Q0)이다. 표 1은 FNPM의 19개의 제어규칙을 보이는 것으로, 하위 우선순위들의 전송지연에 다소 희생이 있더라도 최상위 우선순위인 우선순위6의 전송지연을 가급적 빠른 시간내에 요구 전송지연 시간 이내로 수렴하도록 한다. 그리고, 우선순위6의 지연이 요구 전송지연 범위안에 해당되면, 다음 우선순위 순으로 각각의 요구 전송지연을 만족시킬 수 있도록 자신의 전송지연을 관리하도록 하였다. 특히 규칙 15에서 17까지는 TRT0가 통신을 위한 최저 한계인 121  $\mu$ sec보다 짧을 때에만 작동하도록 고안하였다. 이상은 퍼지화부와 비퍼지화부에 선정된 값과 마찬가지로 많은 실험을 통해 선정된 것이다.

III. GA에 의한 퍼지 소속함수의 자기구성

1. 소속함수 자기구성을 위한 접근방법  
네트워크 관리자의 관리 능력이나 주관적인 경험과

표 1. 퍼지 네트워크 성능관리기의 제어규칙(15-17 번 규칙 : TRT0가 121 μsec일 때 적용).

Table 1. Fuzzy rules for FNPM(#15-17 : applied as TRT0 equals 121 μ sec).

	TCT	D6	D4	D2	D0	ΔTHT	ΔTRT4	ΔTRT2	ΔTRT0	ΔQ0
1	S					ZR	NB	NB	NB	
2	M	B				PB	NB	NB	NB	
3	M	M				PS	NS	NS	NS	
4	M	S	B			ZR	PB	NS	NS	
5	M	S	M			ZR	PS	ZR	ZR	
6	M	S	S	B		ZR	ZR	PB	NS	
7	M	S	S	M		ZR	ZR	PS	ZR	
8	M	S	S	S	B	ZR	ZR	ZR	PB	
9	M	S	S	S	M	ZR	ZR	ZR	PS	
10	M	XS				NS	PB	NS	NS	
11	M	S	XS			ZR	NS	PB	NS	
12	M	S	S	XS		ZR	ZR	NS	PB	
13	M	S	S	S	S	ZR	ZR	ZR	NS	
14	M	S	S	S	XS	ZR	ZR	NS	PB	
15	M	B								NB
16	M	M								NB
17	M	S								ZR
18	L	S				ZR	PB	PB	PB	
19	L	M				PB	ZR	ZR	ZR	

지식에 의존하는 퍼지 시스템을 보다 객관적으로 관리할 수 있도록 하기 위하여, FNPM에 네트워크 조건을 고려한 소속함수의 자동적인 튜닝 기능과 자기 구성 능력을 부여할 필요가 있다. 이를 위해, 휴리스틱(heuristic) 알고리즘의 일종인 유전자 알고리즘이 적용되었는데, 이것은 전체의 탐색 공간상에서 여러 개의 스트링(string)을 가지고 동시에 탐색을 수행하기 때문에, 전역 최적값으로 수렴할 가능성이 크며, 미분 불가능이나 비선형, 멀티모달(multi-modal)과 같은 어려운 문제에도 강인한(robust) 특성을 발휘한다[12][13].

FNPM의 퍼지화부와 비퍼지화부의 소속함수를 자기 구성하기 위한 유전자 알고리즘의 적용방식은 그림 3과 같으며, 그 과정은 다음과 같다.

① 정의된 랜덤발생기에 의해 초기 타이머 값과 초기 모집단을 생성한다. 여기서 초기 모집단은 퍼지부와 비퍼지부의 소속함수에 대한 이진 스트링을 의미하는 것이며, 초기 타이머는 IEEE 802.4 토큰버스 프로토콜의 우선순위 규정에 부합하도록 생성시킨다.

② 생성된 초기 타이머는 토큰버스 시뮬레이션 모델의 입력으로서, 각 우선순위들의 전송지연과 throughput 등과 같은 정보를 시뮬레이션을 통하여 추출하고, 이를 FNPM에 입력한다.

③ 초기 모집단인 FNPM의 지식베이스부는 개체에 따라 각기 다른 퍼지화부와 비퍼지화부의 적용에 의해 새로운 타이머 값들이 인구(population) 수 만큼 생성되어 다시 토큰버스 시뮬레이션 모델에 적용된다

④ 시뮬레이션 모델을 통해 산출된 각 스트링들은 적합도 함수에 의해 평가(evaluation) 된다

⑤ 각 스트링들은 재생산과 교배, 돌연변이를 거쳐 새

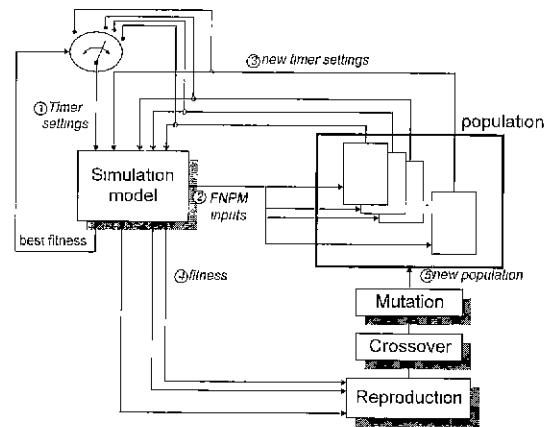


그림 3. FNPM 지식베이스 자기구성을 위한 GA의 적용

Fig. 3. Configuration for self-organization of FNPM knowledge base.

표 2 GA 파라미터 조건.

Table 2. GA parameter condition.

		P6	P4	P2	P0	t0	ΔQ0
population size		30					
crossover probability		0.9					
mutation probability		0.01					
string length (bit)	input variables	36	39	45	57	24	
	output variables	32	32	32	32		12
fixed points for input variables (where "Small" point take the membership of i)		6500	12000	30000	5E5	0.5	
output variable limits (μ sec)	lower	-400	-400	-500	-600		-12
	upper	500	500	400	350		12

로운 개체군을 형성함으로써 진화한다.

이상과 같은 과정은 제한된 세대에 도달하거나, 적합도의 향상 정도가 정해진 값 이하가 될 때까지 계속해서 반복하게 되며, 새로운 개체군이 부모 세대보다 나쁜 결과를 발생시키는 것을 방지하기 위하여 엘리트스트 모델(elitist model)을 적용하였다. 또, 여러 가지 변수들이 공존하여 복합적으로 작용하는 경우에 사용하는 다변수 코딩(multiparameter coding)을 사용하여, 교배와 돌연변이가 고정된 각 변수 영역들을 하나의 서브-스트링으로 하여 각각의 서브-스트링마다 확률에 따라 교배점과 돌연변이점이 달리 발생하도록 한 것이다.

표 2는 유전자 알고리즘의 변수에 대한 적용 조건을 보이는 것이며, 입출력 변수는 각각 5개이며, 한 스트링의 전체 길이는 그림 4와 같이 341비트이다. 여기서, TCT에 대한 소속함수의 조정은 이 값이 선정된 타이머 값에 대하여 상대적으로 평가되기 때문에 배제되었다. 그리고, 각 서브-스트링의 코딩은 절대값 편차방식을 적용하였는데, 그림 5에 우선순위6에 대한 예를 나타낸다. 이것은 (a)의 퍼지화부 자기구성과 같이, 전송지연 제약조건 (small이 1이 되는 점)이 6500 μsec로 관리자가 미리 설

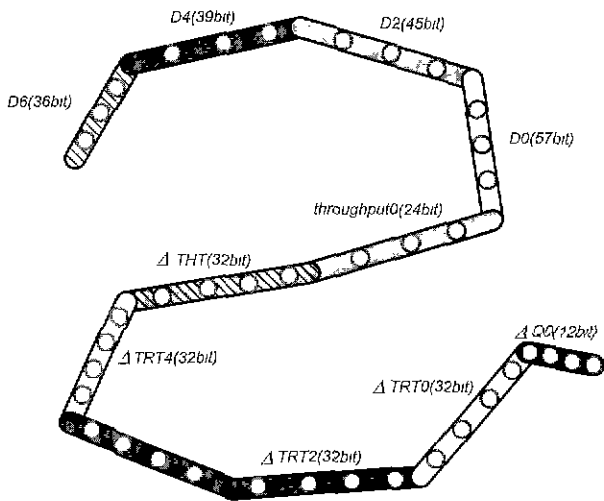


그림 4. 적용된 다변수 코딩.  
Fig. 4. Multiparameter code constructed from concatenated, fixed-point codes.

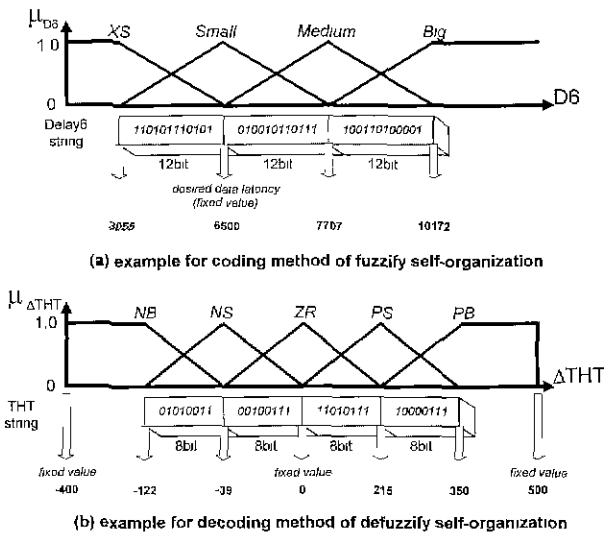


그림 5. 코딩과 디코딩에 있어 우선순위6에 의한 예.  
Fig. 5. Example for coding and decoding of p6.

정한 상태에서 전체 스트링의 첫 번째 서브-스트링 12비트의 십진수 값은 XS의 값(6500 - 3445 = 3055 μsec)을 나타낸다. 그리고, 두 번째 서브-스트링인 13비트부터 24비트까지의 십진수 값은 Medium의 값(6500 + 1207 = 7707 μsec)을 나타내며, 세 번째 서브-스트링인 25비트부터 36비트까지의 십진수 값은 Big의 값(7707 + 2465 = 10172 μsec)을 의미한다. 또한, (b)의 비퍼지화부 자기구성은 관리자가 최대한의 비퍼지화 영역을 미리 설정한 값(-400과 0, 500 μsec)을 기준으로 각 8비트 단위로 (a)의 경우와 같이 수행된다. 여기서, 만약 NB나 PB 등이 각각 관리자가 설정한 최소, 최대의 값을 초과하면 모든 언어 변수 영역값을 동등한 비율로 스케일링(Scaling)하게 된다.

그리고 우선순위4, 2, 0에 대해서는 표 2와 같이 12000, 30000, 50000 μsec을 기준으로 각각 13, 15, 19비트의

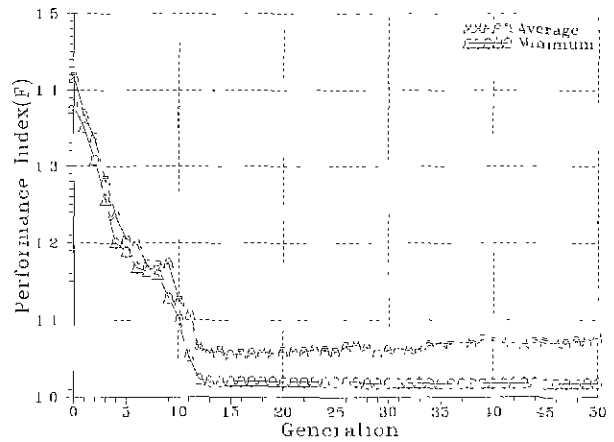


그림 6. 소속함수 자기구성을 위한 성능지수.  
Fig. 6. Performance index by self-organization of membership functions.

크기로 퍼지화부의 소속함수를 표현하며, 우선순위0의 throughput에 대해서는 0.5를 기준으로 8비트로 표현하였다.

적합도 함수는 IEEE 802.4 토큰버스 네트워크에 있어서 전송된 각 메시지의 지연에 근거하여 계산하였으며, 성능지수 F는 (1)과 같다. 여기서, i(6, 4, 2, 0)는 각 우선순위를 나타내며,  $D_i$ 는 우선순위 i의 평균 전송지연을 나타낸다. 그리고,  $R_i$ 는 우선순위 i의 전송지연 제약조건을 의미하는 것이며,  $\alpha$ 와  $\beta$ ,  $\gamma$ 와 같은 상수들은 각 우선순위별 가중치를 나타낸다. 이것은 유전자 알고리즘의 탐색에 있어 성능지수 F를 감소시키는데 상위우선순위의 평균 전송지연을 해당 전송지연 제약조건에 만족시키는 것이 하위우선순위의 전송지연을 크게 감소시키는 것보다 더 효과적이라는 IEEE 802.4 토큰버스 프로토콜을 (1)에 반영한 것이다. 이와 같은 각 우선순위의 상수들은 각 우선순위의 중요도를 반영하는 가중 인자(Weighting factor)로서, 여기서는 각각 0.6, 0.25, 0.1로 설정하였는데, 이것은 여러 가지 연구와 실험결과에 의한 경험치이다.

$$F = \alpha \cdot \frac{D_6}{R_6} + \beta \cdot \frac{D_4}{R_4} + \gamma \cdot \frac{D_2}{R_2} + (1 - \alpha - \beta - \gamma) \cdot \frac{D_0}{R_0} \quad (1)$$

2. 소속함수 자기구성 실험

80%의 트래픽에서 사건이 일어나는 순서와 시간이 매 세대마다 동일한 단일 씨드(seed)를 통해 FNPM의 퍼지화부와 비퍼지화부의 소속함수를 생성시키도록 하였다.

그림 6은 유전자 알고리즘이 성능관리에 적용되어, 스트링들을 진화시켜갈 때의 성능지수 F를 보이는 것으로, 매 세대의 우수 스트링과 그 세대의 전체 스트링의 평균 성능지수 추이는 비슷한 경향을 가짐을 알 수 있다. 여기서, 유전자 알고리즘은 성능지수 F가 감소되도록 적절히 적용하여 초기에 대략 1.38의 성능지수를 12세대 정도에서 1.02로 감소시킴을 볼 수 있다. 그러나, 스트링 길이에 대해 인구 수가 그다지 크지 않아, 다양성(diversity)을 유지하는데 효과적이지 못하여 계속된

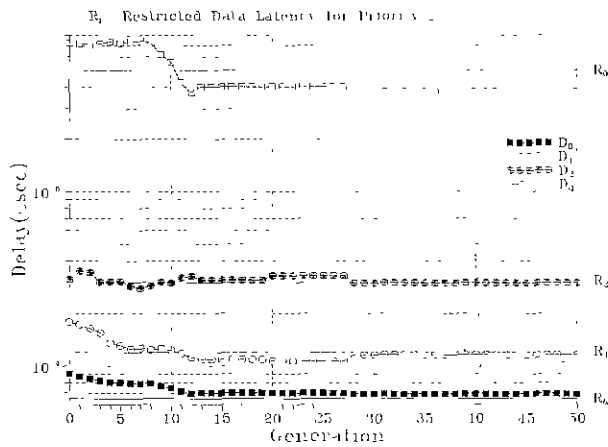


그림 7. 소속함수 자기구성 동안의 최고 우성 스트링에 대한 전송지연.  
 Fig. 7. Average data latencies with the best string for self-organization of membership functions.

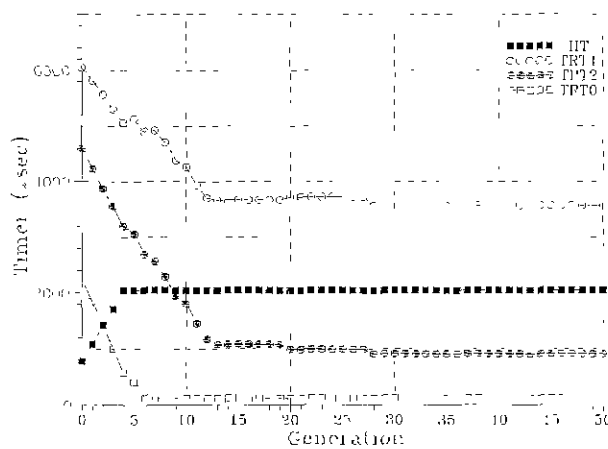


그림 8. 소속함수 자기구성 동안의 최고 우성 스트링에 대한 타이머 변화 추이.  
 Fig. 8. Timer settings with the best string for self-organization of membership functions.

세대의 진전에도 불구하고 더욱 향상된 스트링이 발견되지 않아 더 이상 성능지수가 감소하지 않음을 볼 수 있다.

그림 7은 각 세대에 있어 최고 우성의 스트링에 대한 전송지연을 나타낸 것이며, 그림 8은 이때의 타이머를 보인다. 성능지수에서 가장 큰 가중치를 가지는 우선순위6는 전송지연 제약조건이 6500 μsec로서, 초기의 9000 μsec의 전송지연이 6300 μsec까지 계속해서 감소하였다. 그리고 우선순위4도 18000 μsec에서 전송지연 제약조건인 12000 μsec 근처까지 감소함을 보인다. 우선순위2는 초기에 전송지연 제약조건을 만족하였으나, 상위 우선순위의 전송지연 변화에 따라 자신의 전송지연도 영향을 받기 때문에 타이머를 조정하여 전송지연을 관리하고 있다. 우선순위0는 초기의 70000 μsec라는 전송지연에서 자신의 타이머를 감소시키기도 하지만, TRT4와 2의 감소로 인해 전송지연은 그다지 증가하지 않다가 8세대에서 자신의 큐 용량을 감소시켜 상위 우선순위의 전송

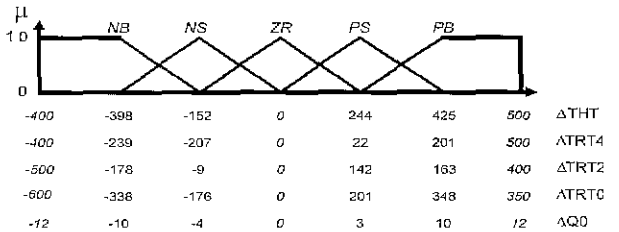
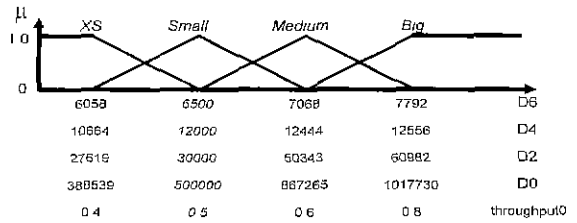


그림 9 GA에 의해 자기구성된 소속함수.  
 Fig. 9. Membership functions by GA self-organization

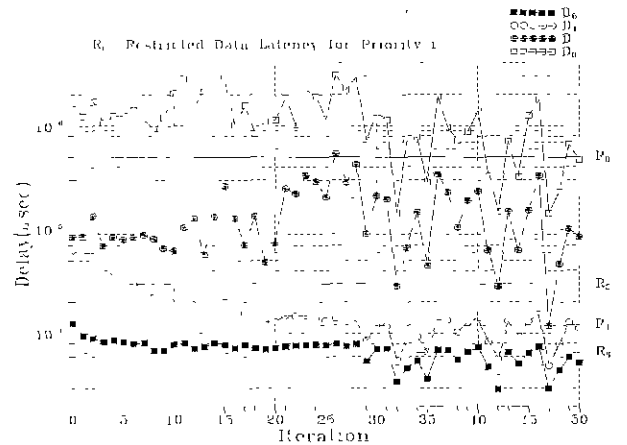


그림 10. 경험의 소속함수에 의한 전송지연.  
 Fig. 10. Average data latencies with membership functions based on human expert knowledge.

지연 감소에는 물론이고, 자신의 전송지연도 감소시킴을 볼 수 있다.

이상과 같이, 유전자 알고리즘의 탐색 결과, 그림 9와 같은 결과를 도출하였으며, 여기서 이탤릭체의 값들은 관리자가 미리 설정한 전송지연 제약조건이다.

3. 관리자의 경험과 GA에 의한 소속함수의 성능 비교

본 실험은 동일한 트래픽에서 사건의 발생 순서와 시간이 각기 다른 51가지의 씨드를 사용하여 수행하였으며, 그림 2와 같이 관리자의 경험을 기초로 구성된 소속함수와 그림 9의 GA 탐색에 의해 자기 구성된 소속함수와의 비교를 통해 자기 구성된 소속함수를 검증하기 위한 것이다.

그림 10은 관리자의 경험에 의한 소속함수를 기초로 하여 수행된 전송지연에 관한 실험 결과를 나타낸 것이다. 우선순위6와 4의 전송지연은 네가지 타이머의 적절한 할당과 큐 용량의 조절로 완만한 감소 추세이기는 하나, 대략 30 iteration에서 전송지연 제약조건을 만족시키

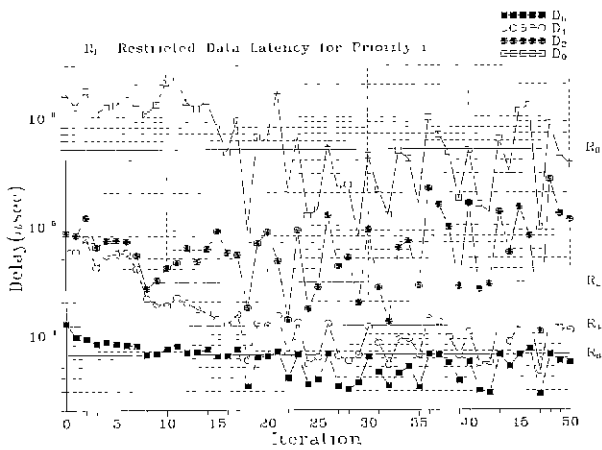


그림 11. 자기구성된 소속함수에 의한 전송지연.  
Fig. 11. Average data latencies with membership functions based on GA self-organization.

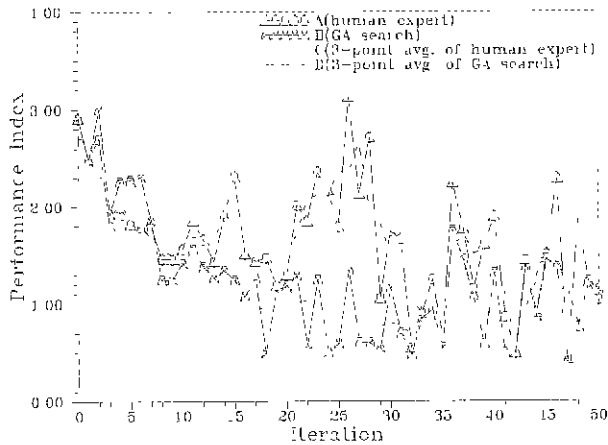


그림 12. 관리자의 경험과 자기구성된 소속함수의 성능 비교.  
Fig. 12. Performance index of human expert and GA search membership function.

고 있다. 또한, 우선순위2와 0의 전송지연은 상위 우선순위의 전송지연을 관리하기 위해 자신의 타이머와 큐를 희생함으로써, iteration 30이전까지는 큰 변화를 보이지 않고 있다. 그러나, iteration 30이후에는 우선순위0의 큐 용량이 삭감되어 전체 트래픽이 감소되고, 이로 인하여 전송지연이 전체적으로 감소하는 것을 보여준다.

한편, GA에 의해 자기 구성된 소속함수를 이용한 각 우선순위의 전송지연을 보이고 있는데, 그림 11은 관리자의 경험에 의한 경우와 매우 흡사하다. 하지만, 상위 우선순위에 대한 배려가 더욱 확고함을 알 수 있으며, 우선순위2를 제외한 모든 우선순위들이 전송지연 제약조건을 만족시키고 있음을 보인다. 이것은 TRT0는 물론이고 TRT2까지도 상위 우선순위를 존중하는 정책(policy)을 반영한 비퍼지화부에 의한 것으로 우선순위6의 전송지연이 전송지연 제약조건을 만족하지 못할 때, 하위 우선순위에 대한 규제가 더욱 엄격했음을 알 수 있다.

그리고, 그림 12는 관리자의 경험에 의한 소속함수(A)

와 GA 탐색에 의한 소속함수(B)를 기초로 얻어진 성능지수에 대한 결과를 비교한 것이다. 성능지수는 (1)에 의한 것이며, 하위 우선순위보다 상위 우선순위의 전송지연을 적극적으로 관리하도록 상수를 구성한 것이다. 여기서 A와 B는 모두 전반적으로 성능지수를 감소시키는 추세를 가지지만, B의 경우가 초기부터 계속해서 더 향상된 결과를 보이는 것을 알 수 있다. 이것은 유전자 알고리즘의 탐색에 의한 소속함수가 상위 우선순위에 대한 배려와 하위 우선순위에 대한 규제를 더 포괄적으로 함축하였으며, 초기부터 Q0를 감소시키기 때문에 20에서 30iteration에서 큰 차이를 보이고 있다. 이는, 우선순위 도구의 목적과 부합하는 형태의 소속함수를 자기 구성한 것이라고 할 수 있다. 그리고 C와 D는 각각 A와 B에 대한 3점 평균으로 씨드의 변동 폭이 큰 점을 감안하여 구성하였다. GA 탐색에 의한 소속함수 적용이 초기부터 성능지수를 더욱 더 감소시킴으로써, 향상된 결과를 보이고 있음을 알 수 있다.

IV. 유전자 알고리즘에 의한 퍼지 규칙의 자기구성

1 퍼지 규칙의 자기구성을 위한 접근방법

퍼지 규칙의 자기 구성을 위한 제어 규칙은 19개로서, 표 1의 짧은 실선어내의 결론부가 하나의 스트링으로 표현된다. 또한, 유전자 알고리즘의 적용 방법도 소속함수의 자기 구성과 마찬가지로 그림 3과 같은 과정을 거친다. 단, 소속함수의 자기 구성에서는 유전자 스트링이 이진수인데 반해, 여기서는 문자형(character type)의 스트링으로, 이러한 문자형 스트링은 랜덤 발생기에 의해 0과 4사이의 값을 발생토록 하여 0인 경우는 NB, 1인 경우는 NS, 2인 경우 ZR, 3인 경우는 PS, 4인 경우는 PB로 구성된다.

효과적인 퍼지 규칙의 생성을 위해, 교배 방식은 면적 교배법(method of area crossover)을 사용하였으며, 이것은 면적의 형태를 가진 염색체에 적용되는 것으로 확률에 따라 임의로 발생된 교배 중심(center of crossover)과 교배 반경(radius of crossover)을 이용하여 정해진 면적 이내에 속하는 형질(allele)을 서로 맞바꾸는 역할을 함으로써 교배가 발생하는 것을 의미한다. 이를 그림 13에 나타내고 있는데, 염색체의 횡축 방향을 각각 X, Y라고 할 때, 교배 중심은 (7,1), 교배 반경은 (2,4)인 경우에 있어 교배 형태를 보이고 있다. 그리고 돌연변이는 임의의 형질을 전환하는 것으로, 만약 선정된 형질의 값이 NB나 NS이면, 랜덤 발생기에서 ZR과 PS, PB 중의 문자형 형질을 임의로 발생시켜 대치하게 된다. 그리고 PS나 PB이면 반대로 ZR나 NS, NB 중의 형질을 선택하고, ZR이면 ZR를 제외한 모든 문자형 형질이 가능성을 가지게 된다.

퍼지 규칙의 자기 구성을 위한 유전자 알고리즘의 파라미터 조건은 표 3과 같으며, 교배 중심의 발생 영역에 있어 횡축(X)상으로는  $1 \leq X \leq 5$ , 종축(Y)상으로는  $1 \leq Y \leq 19$  이내에서 생성될 수 있으며, 교배 반경( $R_x, R_y$ )은  $1 \leq R_x \leq 3, 1 \leq R_y \leq 10$ 으로, 스트링의 형태와 유사한 교배 영역이 발생할 수 있도록 구성하였다. 또한, 돌연변이



표 4. GA에 의해 탐색된 FNPM의 퍼지 규칙.

Table 4. Fuzzy rules for searched FNPM by GA search.

	TCT	D6	D4	D2	D0	ΔTHT	ΔTRT4	ΔTRT2	ΔTRT0	ΔQ0
1	S					ZR	NS	ZR	ZR	
2	M	B				PS	PS	ZR	NS	
3	M	M				ZR	PS	ZR	NS	
4	M	S	B			ZR	ZR	ZR	NS	
5	M	S	M			PS	ZR	ZR	ZR	
6	M	S	S	B		ZR	ZR	PS	ZR	
7	M	S	S	M		ZR	PS	PS	ZR	
8	M	S	S	S	B	PS	PS	ZR	PS	
9	M	S	S	S	M	NS	PS	PB	ZR	
10	M	XS				NS	PS	PS	ZR	
11	M	S	XS			ZR	NS	PS	ZR	
12	M	S	S	XS		PB	PB	PB	NB	
13	M	S	S	S	S	ZR	NS	NS	ZR	
14	M	S	S	S	XS	PS	ZR	NS	PB	
15	M	B								NB
16	M	M								NS
17	M	S								ZR
18	L	S				NS	PS	ZR	NB	
19	L	M				NS	NS	NB	ZR	

였는데, 이것은 해당 형질들에게 퍼지 추론 과정에서 발생하는 min-max 소속값들을 가중시킨 후, 각각을 합산한 뒤 소속값들의 합을 나누는 방식이다. 이것을 (2)에 나타낸다.

$$LP_j(i) = \frac{\sum_{k=1}^n (VLP_k^j(i) \cdot VM_k^j(i))}{\sum_{k=1}^n VM_k^j(i)} \quad (2)$$

여기서,  $i(6,4,2,0)$ 은 우선순위,  $j$ 는 1번에서 19번까지의 해당 규칙의 번호를 의미하며,  $LP_j(i)$ 는  $j$ 번째 규칙의  $i$  우선순위에 대한 결론부의 언어변수(Linguistic Parameter)를 나타낸다. 그리고  $n$ 은 해당 규칙에 대한 탐색 횟수를 의미하며,  $VLP_k^j(i)$ 와  $VM_k^j(i)$ 는  $j$ 번째 규칙에 있어  $i$ 우선순위에 대하여 풀에 저장된  $k$ 번째 언어변수값(Value of Linguistic Parameter)과 그 규칙이 적용된 강도(firing strength)를 나타낸다.  $VLP$ 에 대입되는 값으로 언어변수가 NB이면 0, NS이면 1, ZR이면 2, PS이면 3, PB이면 4를 대입하여, 그 때의 소속값(VM)을 곱한 후 합산하여  $j$ 번째 규칙에 있어  $i$ 우선순위의 VM들을 합하여, 이 값으로 나누는 후, 최종적으로 계산되어진  $LP_j(i)$ 는 가까운 정수값에 따라 언어변수로 복귀된다. 즉, 어떤 세대에서  $j$ 번째 규칙이 0.8의 강도로 ZR, NS, ZR, NB와 0.3의 강도로 PS, NS, PS, PB가 저장되어 있다면, 이것을 통합한 언어변수는 (2)에 의해 2.3, 1, 2.3, 1.1이 되어 각각 ZR, NS, ZR, NS가 되는데, 만약 2.5라면 ZR과 PS 중 랜덤하게 취하게 된다. 여기서, 만약 어떤 규칙이 한 번도 유전자 알고리즘의 탐색에 적용되지 않으면, 최종의 우선 스트링에서 해당하는 규칙의 형질을 복사한다. 이상과 같은 방법에 의해, 최종적으로 탐색된 규칙 배이

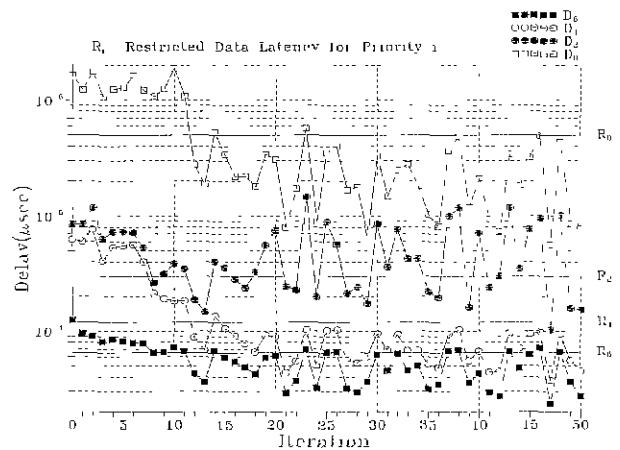


그림 15. 경험에 근거한 퍼지 규칙에 의한 전송지연  
Fig. 15. Data latencies by using fuzzy rule of human expert.

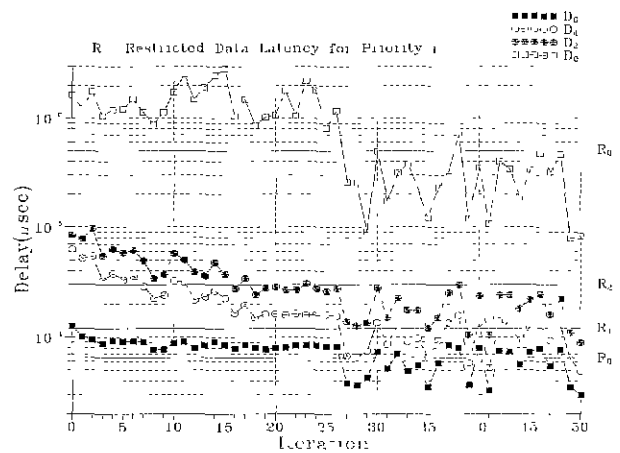


그림 16. 자기구성된 퍼지 규칙에 의한 전송지연.  
Fig. 16. Data latencies by using fuzzy rules from GA search.

스는 표 4와 같다.

### 3. 관리자의 경험과 GA탐색에 의한 퍼지 규칙의 성능 비교

본 실험은 GA가 FNPM의 제어 규칙을 효과적으로 자기 구성할 수 있는가를 확인하기 위한 것이다. 이를 위해 우선적으로 유전자 알고리즘을 이용한 퍼지 규칙을 자기 구성하고, 자기 구성된 퍼지 규칙과 네트워크 관리자가 이미 많은 실험을 통해 획득한 지식을 기초로 선정된 퍼지 규칙을 성능 평가함으로써 비교할 수 있는 근거를 제시한다. 이를 위해, 다중 씨드를 이용하였으며, 퍼지화부와 비퍼지화부의 소속함수는 그림 9와 같이 이미 GA에 의해 탐색된 값을 적용하였다.

그림 15는 관리자의 지식에 의한 퍼지 규칙의 적용 결과이며, 상위 우선순위의 전송지연을 위해 초반부터 하위 우선순위의 타이머를 대폭적으로 삭감하는 정책을 가지고 있어, 빠른 시간내에 큐 용량을 제어할 수 있는 위치까지 도달했다 따라서, 우선순위2를 제외한 대부분의 우선순위들이 전송지연 제약조건을 만족시키고 있음을 보인다. 반면에 GA 탐색에 의한 그림 16은 전체적인



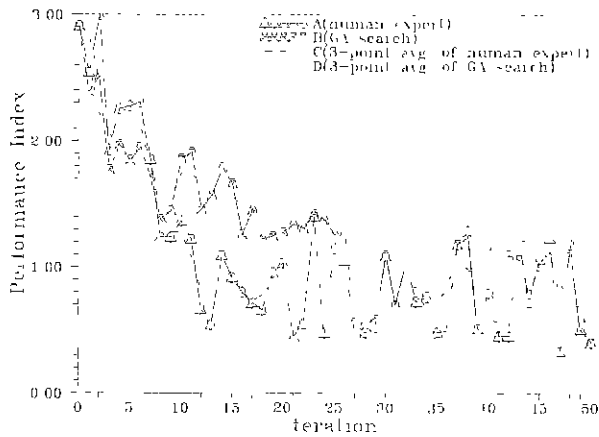


그림 17. 관리자의 경험과 자기구성된 퍼지 규칙의 성능지수 비교

Fig. 17. Performance index of human expert and GA searched rule.

전송지연을 감소시키고 있으나, 상위 우선순위에 대한 정책이 그다지 엄격하지 못해, 우선순위6의 전송지연이 전송지연 제약조건을 만족시키는데 상당한 시간이 걸리고 있다는 것을 알 수 있다. 즉, 상위 우선순위의 전송지연 감소를 위해 하위 우선순위의 타이머를 감소시키는 정책이 그다지 대폭적이지 못하여, 26번의 제어 횟수에 도달해서야 큐 정책을 사용하여 우선순위6의 전송지연을 전송지연 제약조건에 만족시키고 있다 또한, 우선순위0의 큐용량 조절에 의해 우선순위4, 2, 0의 전송지연은 트래픽의 감소로 인해 모든 우선순위가 전송지연 제약조건을 만족시키고 있음을 알 수 있다.

그림 17은 관리자의 경험에 의해 도출된 규칙 베이스

표 5. 입출력 정규화를 위한 제한치.

Table 5. Limit values for normalizing inputs and outputs.

		input parameter		output parameter	
		lowerbound	upperbound	lowerbound	upperbound
traffic (%)	Priority6	0	100		
	Priority4	0	100		
	Priority2	0	100		
	Priority0	0	100		
TIT( $\mu$ sec)		100	2048		
TRT1( $\mu$ sec)		121	8192		
TRT2( $\mu$ sec)		121	8192		
TRT0( $\mu$ sec)		121	8192		
Queue 0		0	50		
TCT( $\mu$ sec)				121	5000
TCT standard deviation( $\mu$ sec)				0	5000
delay ( $\mu$ sec)	Priority6			1000	20000
	Priority4			3000	100000
	Priority2			6000	1000000
	Priority0			50000	4000000

(A)와 GA 탐색에 의해 탐색된 규칙 베이스(B)를 기초로 동일한 조건을 적용함으로써, 얻어진 성능지수에 대한 결과를 비교한 것이다. B의 적용에 의한 실험 결과는 오랜 실험을 통해 얻어진 경험의 결과보다는 성능지수를 감소시키는데 있어 적극적이지 못했는데, 이것은 그림 15에서 나타냈듯이, 타이머에 대한 정책이 A의 경우보다 미온적인 것으로 큐를 관리하는 시점에서 큰 차이를 보이기 때문이다 하지만, 제어 횟수가 증가할수록 두 가지 규칙 베이스에 의한 성능지수는 비슷한 경향을 보이고 있어, 유전자 알고리즘이 탐색한 규칙이 제어 목적과는 상이하지 않음을 알 수 있다 그리고, C와 D는 A와 B에 대한 3점 평균을 보이고 있어, 감소 추세를 효과적으로 나타낸다.

이상의 결과를 살펴보면, GA가 탐색한 표 4의 퍼지 규칙은 우선순위 정책에 있어 다소 미흡한 면을 보이고 있다. 이것은 타이머 정책과 큐 용량에 대한 정책으로 쉽게 알 수 있으며, 더욱 엄격한 하위 우선순위의 타이머 감소량이 요구된다. 하지만, 네트워크의 성능관리에 있어 이와 같은 결과는 많은 잇점을 가지고 있는데, 우선은 관리자의 전반적인 경험이나 지식을 필요로 하지 않으면서, 짧은 시간내에 관리자의 능력과 유사한 결과를 도출할 수 있음을 확인하였고, 또한 미지의 시스템에 대한 정보를 조사하거나 제어하고자 할 때, GA 탐색에 의한 지식 베이스의 자기 구성은 유용한 방법이라고 할 수 있다.

4. 신경망을 이용한 공장자동화용 네트워크의 모사 (approximation) 및 평가

토큰버스 시뮬레이션 모델을 통한 GA의 적용은 FNPM의 지식베이스를 자기 구성하는데 있어, 개체군의 크기와 시뮬레이션 관측 시간 등의 제한 사항이 따랐다. 따라서, 이를 극복하면서 보다 효과적인 탐색을 수행하기 위하여 IEEE 802.4 토큰버스 네트워크의 특성을 신경망에 모사시키는 모델(Neural Network Approximation Model, NNAM)을 구축하여, 이것을 대상으로 GA가 탐색을 수행하도록 하였다.

NNAM의 구조는 그림 18과 같으며, 입력층을 포함하여 세 개의 층으로 구성된다. 또한 은닉층의 뉴런 개수는 100개로 선정하였는데, 이것은 신경망에서 은닉층의 뉴런 개수를 체계적으로 결정하기 위한 많은 연구가 있었으나, 일반적으로 시행착오적으로 최적의 수를 결정하고 있다. 신경망 모델의 입력은 각 우선순위의 트래픽과 프로토콜의 네가지 타이머, 우선순위0의 큐 용량을 포함해 9개이며, 입력 요소는 아니지만 사람의 편견과 같은 역할을 하는 1개의 바이어스(bias)를 두었다. 그리고 출력은 토큰순환시간의 평균과 표준편차, 각 우선순위의 전송지연(평균 전송지연 + 3×전송지연 표준편차)을 포함한 6개의 뉴런이다. 또한, NNAM의 학습 데이터는 여러 가지 입력변수를 달리한 조건하에서 네트워크 시뮬레이션 모델을 수행하여 얻어진 것이다. 산출된 입력과 출력에 대한 네트워크 정보를 트레이닝 셀(training set)으로 구성하여 학습을 수행하는데, 트레이닝 셀에 포함된

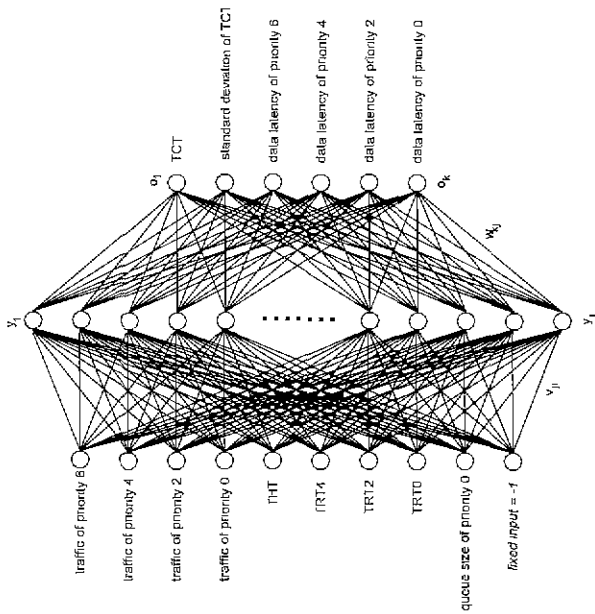


그림 18. NNAM의 구조.  
Fig 18. Configuration of NNAM (Neural Networks Approximation Model).

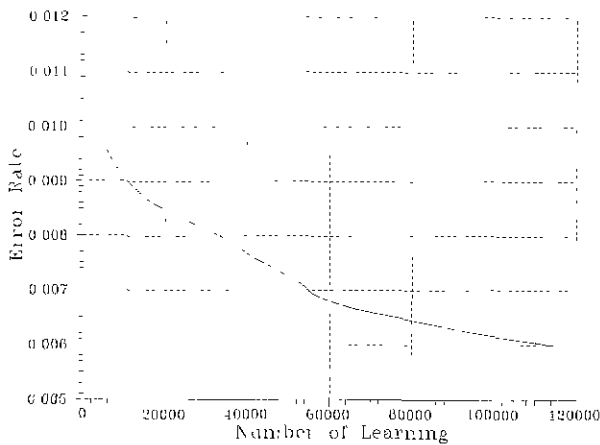


그림 19. 학습 횟수에 대한 평균 에러율.  
Fig. 19. Trend of error rate on learning.

학습 데이터는 600개이다. 이와 같은 학습 데이터는 표 5와 같이 각 입출력값에 대한 최소 한계치와 최대 한계치를 기초로 0에서 1사이로 선형적인 정규화(normalization)가 수행된다.

본 연구에서 적용된 학습 알고리즘은 학습시간 단축에 효과적인 모멘텀 역전파 알고리즘(momentum back propagation algorithm)으로, 이것은 학습률(learning rate,  $\alpha$ )과 각 층의 에러값. 이전 학습 단계의 연결강도 변화량을 추가적으로 적용하기 위한 모멘텀 상수( $\beta$ )에 의해서 각각의 연결강도가 변화한다. 여기서, NNAM에 적용된 학습률과 모멘텀 상수는 시행 착오적(trial and error)으로 선정되었는데, 각각 0.7과 0.5로 선정하였다. 그리고, 각 층의 출력 계산시 단극성의 시그모이드(unipolar sigmoid) 함수에 적용되며, 학습 완료 조건은 정규화된 평균 에러가 0.006이하가 되면 학습을 중지하도록 하였다.

표 6. NNAM 출력 에러율.

Table 6. Percent error of NNAM output neuron

		min error	max error	avg. error
total error	recall	6.0E-5	7.0E-2	7.1E-3
	interpolation	3.8E-5	1.5E-1	8.2E-3
TCT	recall	2.2E-6	3.6E-3	6.1E-4
	interpolation	3.7E-7	3.7E-2	1.7E-3
$\sigma$ of TCT	recall	1.2E-8	2.5E-3	3.9E-4
	interpolation	1.2E-8	1.6E-2	6.7E-4
D6	recall	6.7E-7	1.3E-2	1.2E-3
	interpolation	1.2E-7	1.1E-2	8.5E-4
D4	recall	2.5E-9	3.5E-3	3.5E-4
	interpolation	3.2E-9	1.6E-2	8.0E-4
D2	recall	4.0E-11	1.4E-2	4.8E-4
	interpolation	1.0E-9	3.6E-3	1.3E-4
D0	recall	3E-11	4.6E-2	4.0E-3
	interpolation	1.2E-6	7.3E-2	4.1E-3

이상에 의해 600개의 학습 데이터를 이용하여 토큰버스 네트워크의 특성을 신경망에 학습하였으며, 그림 19는 학습 횟수에 대한 평균 에러율을 보이고 있다. 학습이 진행됨에 따라 평균 에러는 지속적으로 감소되는 것을 알 수 있으며, 초기점은 1000번의 학습이 이루어졌을 때의 평균 에러를 보인다. 또, 평균 에러율이 0.006이하이던 학습을 종료하게 되는데, 그 때의 학습 횟수는 약 114000번이었다.

이상과 같이, 학습된 NNAM을 기초로 네트워크의 재생(recall)능력과 보간(interpolation)능력을 평가하였는데, 재생능력은 학습된 600개의 데이터 중에서 임의로 발췌한 50개의 데이터를 이용하여 실제 네트워크와 NNAM과의 에러를 나타낸 것이다. 그리고, 보간능력은 학습된 데이터가 아닌 50개의 임의의 네트워크 데이터를 기초로 실제 네트워크의 출력과 NNAM에 모사된 출력값을 비교하여 에러를 나타냄으로써 비교한다. 표 6은 각 출력 뉴런과 이들의 합에 대한 최대, 최소, 평균 에러율을 나타낸 것으로, 전반적으로 재생능력이 보간능력보다 우수한 것으로 나타났으며, 보간능력도 에러가 그다지 많지 않아 네트워크의 우선순위 특성에 부합하고 있음을 알 수 있다

### V. 통합 네트워크 성능관리기(integrated network performance manager)

#### 1. 통합 네트워크 성능관리기의 구성

이상의 네트워크 성능관리를 위한 알고리즘은 그림 20과 같이 상호간에 복합적으로 결합하여 운용되는데, 이를 통합 네트워크 성능관리기(Integrated Network Performance Manager, INPM)라고 한다. 그림에서, 검은색의 화살표는 온-라인의 상황을 의미하고 흰색의 화살표는 오프-라인을 나타내며, 각각의 구성요소는 다음과 같다.

① 일정 기간, 컴퓨터 네트워크의 전송지연 특성을 관찰하면서 전송지연 제약조건을 만족시키기 위해 프로토콜

파라미터들을 적절히 조절하는 기능을 수행하는 FNPM

② 변화하는 환경에서 FNPM의 지식베이스를 최적의 상태로 자기 구성하기 위한 GA 최적화기(optimizer)

③ 효과적인 GA 탐색을 위해 IEEE 802.4 토큰버스 프로토콜의 특성을 모사한 NNAM

이러한 통합 네트워크 성능관리기가 토큰버스 네트워크와 결합하여 운용되는 적용 형태 및 세부적인 절차는 다음과 같다.

Step 1 : 네트워크 관리자가 각 우선순위에 대한 전송 지연 제약조건의 범위를 선정한다

Step 2 : FNPM의 지식베이스인 퍼지화부와 비퍼지화부의 소속함수값 및 후건부의 퍼지 규칙을 초기화한다. 이것은 관리자의 경험이나 지식을 도입할 수도 있지만, 여기서는 랜덤하게 발생시킨 지식베이스를 사용하여 초기화하고, 이후 GA 최적화기에 의해 탐색된 지식베이스로 계속해서 대치된다.

Step 3 : 토큰버스 네트워크의 제어를 위한 입력 변수에 관한 것으로, 각 우선순위의 타이머값과 큐 용량을 선정한다. 초기 타이머는 랜덤 값을 취하여 수행되며, 큐는 정해진 최대 용량이 설정된다. 이후, step 6의 과정에서 발생된 FNPM의 제어값으로 대치되게 된다.

Step 4 : step 3과 같은 토큰버스 네트워크를 운영하고 성능 데이터를 수집한다

Step 5 : 관찰한 네트워크의 결과로부터 발생된 각 우선순위의 전송지연과 TCT 등의 네트워크 데이터를 획득하여 FNPM의 입력이 된다.

Step 6 : FNPM이 수행되는 것으로, 전송지연 제약조건과 네트워크의 입력 데이터를 비교하여 지식베이스를 기초로 새로운 네트워크 제어값이 출력된다. 이것이 step 3의 값으로 대치되어 step 6까지의 과정을 계속하여 반복하여 네트워크 성능관리를 수행하게 된다.

이상은 온-라인으로 수행되어 네트워크의 성능관리를 수행하는 것이며, 이후에는 온-라인의 성능관리 수행 과정에서 퍼지 네트워크의 지식베이스를 최적화하기 위해 오프-라인으로 수행된다.

Step 7 : 토큰버스 네트워크의 특성을 NNAM에 모사시키기 위한 것으로, FNPM의 네트워크 관리 과정에서 발생된 step 3과 step 5의 데이터를 쌍(pair)으로 구성하여 계속해서 저장된다. 이 때 저장된 데이터의 쌍이 정해진 개수가 되면, NNAM이 학습을 수행하는데, 하나의 데이터 쌍에는 네트워크 상에서 일정 시간동안의 메시지 전송에 따른 통계적인 요소가 포함되어 있다. 이후, 학습된 NNAM이 정해진 에러 범위내로 수렴하면, 학습을 중지하고 step 8의 과정을 수행하게 된다.

Step 8 : 학습된 NNAM을 대상으로 최적의 지식베이스를 탐색하기 위해 GA 최적화기가 수행된다. 이때, GA 최적화기는 제어 목적인 퍼지 규칙을 먼저 탐색하며, 이것이 완료되면, 탐색된 퍼지 규칙을 기초로 소속함수를 자기 구성하는 순서를 가진다. 이를 위해, 전송지연 제약조건을 입력으로 하여 적합도 함수가 구성되며, 구축된 지식베이스는 step 2의 지식베이스를 대치하게 된다.

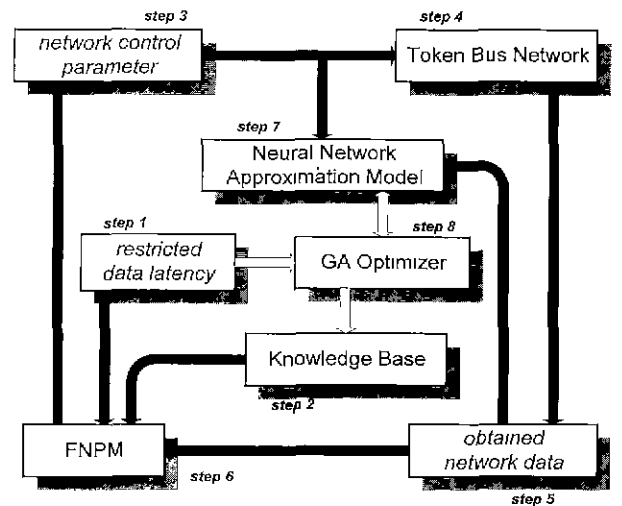


그림 20. INPM의 구조.  
Fig. 20. Configuration of INPM.

이상과 같은 과정을 반복적으로 수행함으로써, INPM은 계속해서 지식베이스를 보강해 나가며, 보다 지능화된 성능관리를 수행할 수 있게 된다.

2. INPM의 운용

INPM은 FNPM이 프로토콜 변수를 제어하는 동안에 발생한 네트워크 데이터를 계속해서 NNAM의 학습을 위한 트레이닝 셀으로 구성한다. 이때, 트레이닝 셀에 포함된 데이터가 120개의 단위가 되면, NNAM은 학습을 수행하게 되며 학습이 완료됨과 동시에 NNAM을 대상으로 전송지연 제약조건을 고려하여 GA 최적화기가 FNPM의 지식베이스를 탐색한다. 이때에도, FNPM의 활동은 계속되어 트레이닝 셀을 보강함으로써, 동적인 네트워크 상황을 보다 NNAM에 효과적으로 모사시킨다.

이와 같은 과정에 의해 생성된 지식베이스의 발달 과정의 예를 보이기 위해, 초기에 120개의 학습 데이터를 모사하여 지식베이스를 탐색한 경우와 240개, 360개, 480개, 600개, 720개, 840개, 960개중 120개, 600개, 960개에 의한 경우를 선별하여 비교한다. 표 7은 120개와 600개, 960개의 데이터를 모사한 신경망을 기초로 탐색된 퍼지 규칙이다. 퍼지 규칙이 탐색되면, 이것을 기초로 소속함수를 찾게 되는데, 자기 구성된 소속함수는 그림 21과 같다

이상과 같이 FNPM은 GA 최적화기에 의해 지식베이스가 자기 구성되며, 자기 구성된 지식베이스를 비교하기 위해 동일한 전송지연 제약조건과 초기 타이머, 큐용량을 기초로 성능 평가가 수행되었다. 그림 22는 학습 데이터의 개수가 다른 NNAM을 대상으로 지식베이스를 탐색하여, 이를 기초로 퍼지 네트워크 성능관리기가 각각 프로토콜 변수를 조절하여 얻어진 성능지수를 3점 평균한 것이다. 120개의 데이터에 의한 지식베이스는 각 우선순위의 전송지연을 효과적으로 감소시키지 못해 성능지수에 그다지 큰 변화가 없다. 이것은 120개의 데이터로는 네트워크의 특성을 모사시키는 것이 어렵다는 것을 나타내며, 600개와 960개의 데이터에 의해 탐색된 지

표 7. 자기 구성된 퍼지 제어 규칙.

Table 7. Fuzzy rules for searched FNPM.

	TCT	D6	D4	D2	D0	ΔTHT			ΔTRT4			ΔTRT2			ΔTRT0			ΔQ0		
						120	600	960	120	600	960	120	600	960	120	600	960	120	600	960
1	S					NS	ZR	ZR	PS	ZR	ZR	NS	PS	ZR	ZR	ZR	NS			
2	M	B				ZR	NS	PB	NS	ZR	ZR	ZR	ZR	NS	NS	NS	NB			
3	M	M				ZR	ZR	PS	PS	NS	PS	NS	PS	NS	PS	NS	NB			
4	M	S	B			PS	ZR	ZR	ZR	ZR	ZR	ZR	ZR	NS	ZR	NS	NS			
5	M	S	M			PS	ZR	ZR	PS	NS	PB	ZR	PS	PS	PS	ZR				
6	M	S	S	B		NS	ZR	PS	NB	NB	ZR	PS	ZR	NS	NS	PB	PS			
7	M	S	S	M		ZR	PS	ZR	PB	NS	ZR	ZR	ZR	ZR	NS	PS	NS			
8	M	S	S	S	B	NS	NS	ZR	NS	ZR	NS	NS	ZR	NS	PB	PS	ZR			
9	M	S	S	S	M	NB	ZR	NS	PB	NS	ZR	ZR	PS	PB	ZR	PS	ZR			
10	M	XS				ZR	ZR	ZR	NB	PS	PS	NS	ZR	NS	NS	PB	PS			
11	M	S	XS			ZR	PS	ZR	NS	NS	NB	PB	ZR	PB	NS	ZR	ZR			
12	M	S	S	XS		PS	NS	ZR	ZR	NS	NS	NS	ZR	NS	PB	PB	PS			
13	M	S	S	S	S	ZR	NB	PS	ZR	PS	PS	NS	NS	NS	NS	ZR	ZR			
14	M	S	S	S	XS	ZR	ZR	ZR	PS	PB	NS	NS	NS	ZR	NS	PB	PB			
15	M	B																ZR	NB	NB
16	M	M																ZR	PS	ZR
17	M	S																PS	ZR	ZR
18	L	S				PS	NS	ZR	PS	ZR	PB	PB	PS	PB	PS	NS	ZR			
19	L	M				NB	PS	NB	PB	ZR	PB	NS	NS	NB	NS	NS	NS			

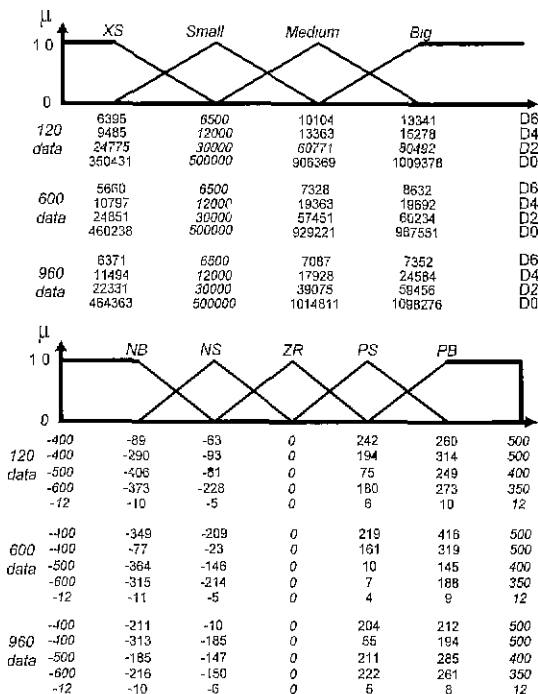


그림 21. NNAM의 학습데이터 개수에 따른 FNPM의 소속함수.

Fig. 21. Membership function for searched FNPM by learning data.

식베이스는 성능관리에 있어 효과적임을 알 수 있었다. 특히, 600개에 의한 경우는 960에 의한 경우보다 타이머나 큐 용량에 대한 정책이 엄격하지 못하여 성능지수의

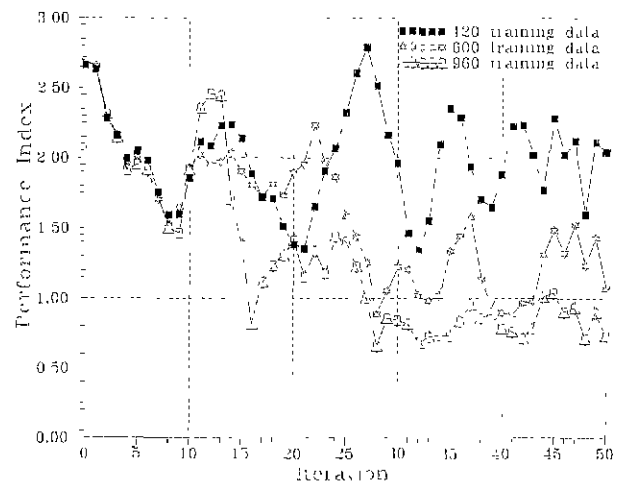


그림 22. NNAM에 의해 자기구성된 지식베이스를 이용한 성능지수 비교.

Fig. 22. Trends of performance index on searched knowledge-bases by NNAMs.

감소 추세가 느린 것을 볼 수 있다. 960개의 학습데이터를 이용한 NNAM이 최적의 학습데이터에 의한 네트워크 모사라고는 할 수 없지만, 학습데이터가 증가하면 보다 효과적인 네트워크 모사가 가능함을 알 수 있었다. 그러나, 이러한 학습데이터의 증가는 은닉층 뉴론수와 같은 신경망 구조와 직접적인 관련이 있는 것으로, 학습데이터를 무한정 증가시킬 수는 없다. 따라서, 제한된 신경망 구조에서 최적의 학습데이터 개수를 구하는 연구도 필요할 것이다.

## VI. 결론

본 논문은 공장자동화용 컴퓨터 네트워크의 성능관리를 위한 것으로, 퍼지 이론과 유전자 알고리즘, 신경망 등의 지능기법을 복합적으로 이용하여 통합된 네트워크 성능관리를 개발하였으며, 시뮬레이션 모델을 통해 다음과 같은 결론을 얻을 수 있었다.

1) 퍼지 네트워크 성능관리기의 지식베이스를 자기 구성하기 위해 유전자 알고리즘을 적용하였으며, 탐색된 소속함수가 경험에 의한 소속함수와 비교하여 상위우선순위에 대한 배려와 하위 우선순위에 대한 규제를 더 포괄적으로 함축하였음을 확인하였다. 또한, 퍼지 규칙의 자기구성에 있어서도 경험에 의한 경우와 유사한 결과를 얻음으로써, 네트워크 관리자의 경험이나 지식없이 짧은 시간내에 관리자의 전반적인 경험과 유사한 결과를 도출할 수 있음을 확인하였다.

2) 9개의 입력 뉴런과 6개의 출력 뉴런을 갖는 NNAM은 단지 600개의 데이터 셀 만을 가지고도 토큰버스 시뮬레이션 모델의 넓은 영역의 특성과 유사함을 보였으며, 이를 재생 능력과 보간 능력에 대한 실험으로 검증하였으며, FNPM이 시뮬레이션 모델을 대상으로 온-라인의 성능관리를 수행하는 도중에도, 유전자 알고리즘이 모사된 신경망을 대상으로 오프-라인에 의해 FNPM의 지식베이스를 향상시킬 수 있는 효과적인 수단을 제시했다.

3) 통합된 네트워크 성능관리기(INPM)는 FNPM과 GA 최적화기, NNAM이 복합적으로 결합하여 지식베이스를 계속해서 향상시키는 네트워크 성능관리기로서, 트레이닝 셀이 많을수록 학습을 위한 시간은 증가하지만, 이를 기초로 유전자 알고리즘이 탐색한 지식베이스는 더욱 효과적이었다.

수행된 연구에서는 대체로 일정한 트래픽이 적용되어 지는 경우의 산업용 네트워크에 대한 것으로, 향후에는 네트워크의 전반적인 트래픽에 대한 지식베이스를 획득함과 동시에 우선순위별로 광범위한 트래픽에 대해서도 검증할 필요가 있다. 또한, 유전자 알고리즘에 의한 지식베이스 자기구성에 있어, 상위 우선순위에 해당하는 메시지가 기각(reject)되는 경우, 기각되는 메시지 수와 하위 우선순위의 throughput 등을 고려할 수 있는 보다 다양한 목적 함수의 구성이 있어야겠으며, 은닉층의 뉴런 수와 학습데이터 개수의 유기적인 관련성과 성능과의 상호 관련에 대한 연구도 수행되어야겠다.

## 참고문헌

- [1] J. T. O'Rourke, "A case for computer integrated manufacturing," *Keynote Address, NSF Workshop on Computer Networking for Manufacturing System*, pp. 5-20, 1987.
- [2] A. Ray, "Networking for computer-integrated manufacturing," *IEEE Network*, vol. 2, no. 3, pp. 40-47, 1988.
- [3] W. Stallings, *Data and Computer Communications*, 4th ed., McMillan, 1994.
- [4] W. L. Genter and K. S. Vastola, "Performance of the token bus for time critical messages in a manufacturing environment," *American Control Conference*, pp. 534-539, 1989.
- [5] S. Lee and A. Ray, "Performance management of multiple access communication networks," *IEEE Jour. Selected Areas in Communication*, vol. 11, no. 9, pp. 1426-1437, 1993.
- [6] 홍승호, "MAP: 공장자동화를 위한 네트워크의 표준," 대한기계학회, 제35권, 제5호, pp. 427-441, 1995.
- [7] S.-H. Lee, J.-W. Son, S. Lee, "Fuzzy performance management of IEEE 802.4 token bus networks," *American Control Conference*, pp. 3254-3258, June, 1995.
- [8] 이상호, 손준우, 이석, "생산자동화를 위한 토큰버스 네트워크의 성능관리," 제어·자동화·시스템 공학, 제2권, 제4호, pp. 42-50, July, 1996.
- [9] 손준우, 이상호, 이석, "뉴로-퍼지 추론 알고리즘에 의한 토큰버스 네트워크의 성능관리," '95 한국자동제어 학술회의 논문집, pp. 1101-1104, Oct., 1995.
- [10] C. D. Pegden, *Introduction to SIMAN*, Systems Modeling Corp., 1985.
- [11] C. D. Pegden, R. E. Shannon, R. P. Sadowski., *Introduction to Simulation Using SIMAN*, McGraw Hill, 1991.
- [12] M. Srinivas and L. M. Patnaik, "Genetic algorithms A survey," *IEEE Computer*, pp. 17-26, June, 1994.
- [13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
- [14] J. M. Zurada, *Introduction to Artificial Neural Network Systems*, West Pub. Co., 1992.
- [15] R. J. T. Morris and B. Samadi, "Neural network control of communications system," *IEEE Trans. Neural Networks*, vol. 5, no. 4, pp. 639-685, 1993.
- [16] F. S. M. Nobre, "Genetic-neuro-fuzzy systems: A promising fusion," *FUZZ-IEEE/IFES '95*, vol. 1, pp. 259-266, 1995.

[1] J. T. O'Rourke, "A case for computer integrated



**이 상 호**

1992년 부산대 생산기계공학과 졸업. 동대학원 석사(1994년), 동대학원 박사(1997년), 1996년-현재 국방과학연구소 제1연구개발본부 제직, 관심분야는 자동화네트워크, 차량 자율주행, 차량성능 시험.



**김 인 준**

1996년 부산대 생산기계공학과 졸업. 동대학원 석사(1998), 1998년-현재 대우정밀공업주식회사 기술연구소 근무. 관심분야는 생산자동화, Field Bus, 차량 세시 시스템.



**이 경 창**

1996년 부산대 생산기계공학과 졸업. 동대학원 석사(1998), 1998년-현재 동대학원 박사과정, 1998년-현재 기계공학연구정보센터 근무. 관심분야는 자동화네트워크, WWW, 데이터베이스.



**이 석**

1984년 서울대 기계공학과 졸업. Pennsylvania State Univ. 석사. (1985), 동대학원 박사(1990년), 1990년-1993년 신시내터 대학교. 기계공학과 조교수 1993년-현재 부산대학교 기계공학부 조교수. 관심분야 자동화네트워크, DES, RP, 자율주행.