분산 Multi-Agent 시스템의 효율적 운용을 위한 신경스케쥴링 기법

이 기 준† · 정 종 필†† · 정 채 영†††

요 약

분산 에이전트 시스템은 에이전트간의 상호협력과 협동을 통하여 주어진 문제의 해를 찾아가지만 개념적으로 매우 복잡하고 에이전트간의 상호작용이 복잡한 시스템이다. 본 논문에서는 이러한 문제들의 해결방안으로 각 에이전트간 효율적인 작업연계를 위한 신경스케쥴링 기법을 제시하였다

먼저 신경스케쥴링 작성을 위하여 시스템운용 시나리오를 바탕으로 총 8750건의 패턴을 구성하였고, 이중 2000건을 학습데이터로 나머지 6750건을 테스트 패턴으로 사용하여 6×5×2의 신경망구조에서 학습하였다 학습결과 학습회수 200회에서 학습패턴과 테스트패턴에 대하여 모두 99%이상의 수렴률을 보였으며, 학습결과를 신경스케쥴링을 위한 데이터로 사용하여 에이전트의 생성, 작업의 수행, 분할 및 상호협력과정에서 잘 수행함을 보였다.

A Neural Scheduling Method for Efficient Use of Distributed Multi-Agent System

Kee-Jun Lee† · Jong-Pil Jeong†† · Chai-Yeoung Jung†††

ABSTRACT

The distributed agent system finds the solution of the problem given through the reciprocal cooperation between agents, but its concept and their mutual operation are complicated This paper suggests a neural scheduling method for an efficient job link between the agents as a solution for these problems First, a total of 8750 case patterns are composed on the base of system operation scenario for drawing up of a neural scheduling. Among these cases, we learned in 6×5×2 neural network by using 2,000 and 6750 cases for learning data and the test pattern respectively. The result of the learning showed more than 90% of convergence rates about learning and test patterns in 200 learning times. The agent creation, its job performance, and the division and mutual cooperation between them were also made well by using the result for a neural scheduling data.

1. 서 론

정보통신기술의 발전은 컴퓨터 이용환경에 커다란 변화를 가져왔다. 컴퓨터의 대량공급과 초고속 유무선 통신망의 확충, 대규모 네트워크 환경, 통신규약의 개방화에 따라 대량의 정보활용이 가능하게 되었다. 이러한 대규모 네트워킹 및 멀티미디어 기술의 흐름은 사용자가 원하는 정보를 원하는 시간안에 다양한 형태로 얻을 수 있도록 제공하였지만, 다른 한편으로는 많은 정보자원을 효율적으로 관리, 활용에 대한 큰 과제를 남겼다. 따라서 이러한 변화에 따른 해결방법으로 분산문제해결(Distributed Problem Solving)기법들이 모색되어 왔으며[1], 이제는 이를 기술적 방법과 경험을 바탕으로 실용화시키기에 이르렀다[2, 9]

본 논문에서는 이러한 분산문제의 주요수단으로 제

† 정 회 원 : 조선대학교 대학원 전산통계학과
†† 정 회 원 : 순천청암대학 컴퓨터정보과학부 교수
††† 종신회원 : 조선대학교 전산통계학과 교수
논문접수 2000년 7월 1일, 심사완료 : 2000년 10월 2일

안되고 있는 분산에이전트 시스템에서 각 Agent 간의 효율적인 작업스케쥴링을 위한 신경스케쥴링 기법을 제안하고자 한다.

스케쥴링 문제는 전통적으로 산업공학등에서 중요한 연구분야였으며, 최적의 스케쥴을 찾는 알고리즘에 대한 많은 연구가 진행되어 왔다. 최근에는 simulated annealing(SA)[3]과 유전자 알고리즘(GA)[4]등 확률적 검색알고리즘을 사용하여 NP-Complete 문제를 해결하려는 시도가 제시되었다. 특히 실제 현장에서 발생하는 스케쥴링문제들은 많은 제약조건과 multi-objective로 인해, 기존의 branch-and-bound, 정수기획법과 같은 exact search방법으로 다루기에는 많은 난점들이 존재했다[5].

이에 비해 신경망 기법은 가중치를 갖는 연결선을 이용하여 상호 작용을 하는 PE(Processing Elements)가 병렬로 구성된 시스템으로 고도의 병렬 처리, 적응성, 학습에 의한 일반화 능력, 결함 허용, 강건성(Robustness), 학습 능력 등의 기능을 지니고 있다[6]. 따라서 에이전트시스템과 같이각 에이전트간의 상호작용이 많은 시스템에서 시스템의 상황, 에이전트의 작업내용 및 사용여부등의 조건에 따른 해당 에이전트와 작업내용을 결정지을 수 있는 이점을 지니고 있다.

제안한 신경스케쥴링(Neural Scheduling)은 분산 에이전트 시스템 운용에 대한 시나리오를 바탕으로 사용자의 우선순위, 현 작업을 수행중인 에이전트이름, 목적에이전트와 사용 가능여부, 목적에이전트에 의해 수행될 작업에이전트, 시스템 자원의 사용가능량을 학습데이터로 사용하고 수행될 작업에이전트와 작업내용을 교사데이터로 사용하여 총 7개의 필드로 구성된 레코드단위의 총 8750건의 패턴을 구성하였다.

총 시나리오 패턴 8750건 중 2000건을 학습데이터로, 나머지 6750건을 테스트패턴으로 사용하여 6×5×2의 신경망 구조에서 학습하였고, 학습결과를 신경스케쥴링 데이터로 사용한다.

본 논문은 2장에서 분산에이전트 시스템의 구성과 작업상황에 대한 시나리오를 작성하고 3장에서는 작성된 시나리오를 바탕으로한 신경스케쥴링 구축방법을 기술한다. 4장에서는 학습된 신경스케쥴링의 성능과 유용성을 평가하고 마지막 5장에서 결론을 맺는다.

2. 분산 Multi-Agent 시스템

분산환경시스템의 예를 설명하기 위하여 여러 작업상황 중 웹(Web) 상에서의 온라인 검색시스템을 운영할 때 발생되는 상황을 바탕으로 하여 분산환경을 알아보도록 하겠다[10]. 온라인 검색시스템은 사용자가 원하는 항목에 대하여 User의 관심사항, 검색패턴을 파악하여 사용자에게 정보를 제공하는 시스템이다. 여기서는 관리에이전트(MA), 인터페이스 에이전트(IA), 질의어 검색에이전트(QA), User 관리에이전트(UA), 사용자 모니터링 에이전트(UMA)등 크게 5가지의 에이전트가 사용된다[7].

2.1 구성 에이전트의 기능

2.1.1 인터페이스 에이전트(IA)

인터페이스 에이전트는 사용자로부터 입력받은 데이터를 관리자 에이전트를 통하여 질의어검색 에이전트에게 보내고, 검색된 결과를 사용자에게 보여주어야 한다.(사용자 인터페이스 기능)

2.2.2 질의어검색 에이전트(QA)

질의어 검색에이전트는 사용자가 입력한 검색단어와 사용자 특징정보를 이용하여 질의문을 구성(질의문 작성기능)하여야 하고, 작성된 질의문을 이용하여 논문DB에 수록된 데이터를 질의하여야 한다(DB 검색기능). 또한 질의한 후 결과를 인터페이스 에이전트에 전달(결과전송기능)과 함께 질의결과에 대한 유사도를 측정하여 검색의 정확도를 나타내어야 한다(유사도 측정기능).

2.2.3 USER 관리 에이전트(UA)

USER 관리 에이전트는 사용자의 특징정보를 저장한 사용자 DB를 관리 수행하여야 한다.(사용자 DB관리 기능)

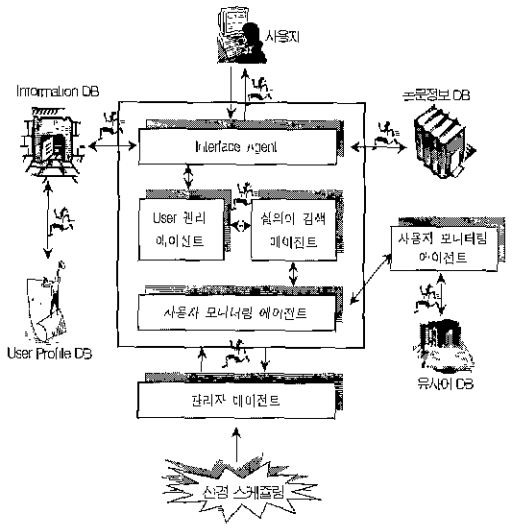
2.2.4 사용자 모니터링 에이전트(UMA)

사용자 모니터링 에이전트는 사용자의 작업사항을 모니터링하여 특성정보를 추출(모니터링 기능)하고, 유사어DB의 관리 작업(유사어DB관리)을 수행해야 한다.

2.2.5 관리자 에이전트(MA)

관리자 에이전트는 시스템의 운영과 각 에이전트에게 작업현황에 맞도록 학습되어진 신경모듈을 이용하여 작업을 분배(신경스케쥴링 기능)한다. 또한 각 에이전트에게 RFP를 전달하고 이에 대한 제안서를 받아냄(통신기능)과 동시에 작업수행을 위해 전달받은 제안

서를 분석하여 실제 작업을 수행한 에이전트는 선택(분석기능)하여야 한다. 그리고 작업수행 시 필요한 에이전트를 생성(복제기능)하고 각 에이전트의 작업수행 요청을 위한 통신패킷을 해당에이전트에게 전송(통신패킷의 전송)하여야 한다.



(그림 1) 멀티에이전트 시스템의 구조

2.2 에이전트의 상호협력 시나리오

멀티에이전트 시스템(Multi-Agent System) 상에 존재하는 여러 에이전트들의 작업 사이에 발생될 수 있는 시스템 사용 자원 여부, 대상 에이전트의 사용 여부, 효율적인 작업의 연계 등의 상황에 따라 대처할 수 있는 시나리오를 작성한다.

① 사용자A가 검색시스템에 접속

- 관리자 에이전트(MA)는 시스템의 인터페이스 에이전트(IA)들의 상태 여부를 파악하여 현재 대기(Wait) 상태의 인터페이스 에이전트(IA)를 가동시킨다.
- 모든 인터페이스 에이전트(IA) 상태가 활동(Busy) 중이면 사용자 A의 작업 우선순위에 의하여 최하위 우선순위 작업을 수행 중인 인터페이스 에이전트(IA)를 대기(Wait) 상태로 전환시킨 후 사용자 A의 작업을 수행하도록 한다. 만일 사용자 A가 최하위 우선순위이면 다른 인터페이스 에이전트(IA)가 작업을 마칠 때까지 대기(Wait) 상태가 된다 만일 사용 가능한 시스템의 자원이 있을 경우 인터페이스 에이전트(IA)를 생성(Create)하여 작업을 수행한다.
- 현재 시스템의 자원이 사용자 A의 작업을 수행할 수 없을 경우, 사용자 A의 우선순위에 의해 최하위 인터페이스 에이전트(IA)를 대기(Wait) 시킨 후 사용자 A의 작업을 수행한다.

② 사용자 A가 검색을 위한 검색단어를 입력

- 사용자 인터페이스 에이전트(IA)는 입력된 검색단어를 질의어검색 에이전트(QA)에게 전달하기 위하여 관리자 에이전트(MA)에게 메시지를 통보한다. 관리자 에이전트(MA)는 현 시스템의 작업 우선순위에 따라 사용 가능한 질의어검색 에이전트(QA)를 조사하여 인터페이스 에이전트(IA)의 메시지를 전달한다.
- 작업의 순서는 사용자 A의 작업 우선순위에 따라 결정된다.

③ 질의어검색 에이전트의 질의문 작성

- 질의어검색 에이전트(QA)는 인터페이스 에이전트(IA)로부터 전달받은 질의어를 이용한 질의문 작성을 위해 먼저 사용자 특성정보를 USER관리 에이전트(UA)에게 요구하고, 요구된 메시지는 관리자 에이전트(MA)를 통하여 USER 관리 에이전트(UA)에게 전달된다.
- 전달받은 USER관리 에이전트(UA)는 사용자 특성정보(Profile) DB를 검색하여 현재 사용자의 이용특성 및 관심 분야 등 사용자에 관련된 정보를 질의어검색 에이전트(QA)에게 전달한다.
- 질의어검색 에이전트(QA)는 도착한 정보를 이용하여 질의문을 작성하고 이를 이용하여 데이터베이스에 질의한 후 결과를 인터페이스 에이전트(IA)에게 전달한다
- 작업의 순서는 사용자 A의 작업 우선순위에 따라 결정된다.

④ 검색결과의 출력

- 인터페이스 에이전트(IA)는 관리자 에이전트(MA)를 통하여 전달된 질의결과를 사용자 A에게 출력한다
- 작업의 순서는 사용자 A의 작업 우선순위에 따라 결정된다.

⑤ DB 갱신

- 관리자 에이전트(MA)는 현재 등록된 사용자정보(Profile) DB의 내용을 갱신하기 위하여 사용자의 행동양식을 모니터링중인 사용자 모니터링에이전트(UMA)로부터 정보를 받아 USER관리 에이전트에게 전달한다.
- 사용자 모니터링에이전트(UMA)는 유사어 DB에서 유사어 및 동의어를 검색하고, 이를 관리자 에이전트(MA)를 통하여 질의어검색 에이전트(QA)에게 전달한다. 이때 질의어검색 에이전트는 사용자 모니터링에이전트(UMA)의 전달 내용을 기반으로 Query 질의문을 작성하고, 사용자 모니터링 에이전트(UMA)는 구성 단어의 유사성을 B-트리 구조로 구성하여 새로운 정보(information)에 대한 적절한 배치를 수행한다.
- 작업의 순서는 사용자 A의 작업 우선순위에 따라 결정된다.

3. 신경 스케줄링(Neural Scheduling)

신경스케줄링(Neural Scheduling) 기법을 이용하여 멀티에이전트 시스템(Multi-Agent System)에서 발생할 수 있는 다양한 상황들에 대하여 적절한 대처의 최적의 에이전트 활용방안을 제시하고자 한다.

3.1 시나리오의 패턴화

위 시나리오에서 기술된 에이전트 사이의 추상적 행동과정들을 신경망을 통하여 학습을 시키기 위해서는 먼저 에이전트의 행동양식에 대한 패턴화가 우선적으로 수행되어야 한다.

먼저 각 에이전트가 수행하는 행위의 종류를 살펴보면 Create, Action, Wait, Wake, Query, Reply, Resear, Connect, Disconnect이 있다. Create 행위는 에이전트의 생성이며, 이때 관리자 에이전트는 시스템에 1개만 존재함으로 시스템 실행시 자동 생성된다 Action 행위는 새로이 생성된 에이전트가 다른 에이전트에 의해 요구되어지거나, 자율적 행위에 의해 작업을 수행할 때 발생되는 행위이다. Wait는 현재 작업에이전트가 작업을 멈추고 대기상태가 되는 것을 의미하며, Wake는 대기상태의 에이전트가 활성화되어 작업이 재 시작되었음을 나타낸다 Query는 목적 에이전트에게 질의문을 보내 어떠한 작업을 의뢰 하는 행위이며, Reply는 Query에 의해 수행된 작업내용을 요구하는 에이전트에게 전달하는 행위이다 그리고 Resear는 관리자 에이전트에 의해 다른 에이전트의 현재 상태 및 활동(busy)여부를 파악한다. Connect는 사용자가 시스템에 접속할 때의 관리자 에이전트의 행위이며, Disconnect는 사용자가 시스템에 접속을 끊을 때 나타나는 행위이다.

패턴화를 위하여 사용자의 우선순위(A1), 현 작업을 수행중인 에이전트들의 이름(A2), 목적에이전트(A3)와 사용 가능 여부(A4), 목적에이전트에 의해 수행될 작업 내용(A5), 시스템의 자원 사용량(A6), 이때 수행될 작업에이전트 이름(B1) 및 작업내용(B2)으로 구성된 레코드 단위의 패턴들을 생성한다. 아래의 <표 1>은 작성된 패턴의 일부내용이다.

<표 1> 에이전트의 패턴적용 예

NUM	A1	A2	A3	A4	A5	A6	B1	B2
1	①	(ㄱ)	(ㅇ)	F	Create	0	(ㅇ)	Create
2	①	(ㄱ)	(ㅇ)	T	Resear	0	(ㅇ)	Wait
3	①	(ㄱ)	(ㅇ)	T	Resear	300	(ㅇ)	Action
…	…	.	.	…	…	…	…	…
243	②	(ㅈ)	(ㅂ)	T	Query	500	(ㄱ)	Resear
244	②	(ㄱ)	(ㅂ)	T	Query	500	(ㅂ)	Query
245	②	(ㅂ)	(ㅅ)	T	Reply	500	(ㄱ)	Replv
246	④	(ㄱ)	(ㅈ)	T	Reply	500	(ㅈ)	Reply
…	.		..	…	..		.	…

위 <표 1>에서 A1 필드는 사용자 우선순위로 ①부터 ⑤사이의 순번을 배정하며, 이 순번은 사용자 특성정보(User Profile) DB에 저장되어 있다. A2 필드는 현 작업을 수행중인 에이전트들의 이름으로 관리자 에이전트는 (ㄱ), 인터페이스 에이전트는 (ㅇ), 질의어검색 에이전트는 (ㅈ), USER관리 에이전트는 (ㅂ), 사용자 모니터링 에이전트는 (ㅁ)로 표기하였다. A3 필드는 작업요청을 수행할 목적 에이전트의 이름으로 A2의 필드와 동일하게 기술하며 만일 목적 에이전트를 필요로 하지 않는 경우 Null로 표기한다 A4 필드는 이때 목적에이전트의 사용 여부로 True와 False로 표기하였다 그리고 A5 필드는 목적에이전트에게 의뢰할 작업의 내용이며, A6 필드는 모든 에이전트의 사용량이 동일하다는 가정 하에 각 에이전트의 실행량은 500으로 설정하고 이때 남은 시스템의 자원을 표기한다.

B1 필드와 B2 필드는 A1~A5 필드의 학습 시 교사데이터로 사용할 필드이다. B1은 작업수행 에이전트

이름으로 A1 필드와 같이 기술하며, B2 필드는 이때 에이전트가 수행할 작업내용이다.

3.2 Descending Epsilon 학습방법[6]

신경망 학습알고리즘은 각 출력노드에서 출력값과 교사데이터의 값의 차가 최소화되도록 연결강도의 값을 조절하는 오류 역전파 학습 알고리즘이 주로 사용된다. 오류 역전파 학습알고리즘은 현재 패턴인식 분야에서 널리 사용되고 있는 학습알고리즘이지만 순수 오류 역전파 알고리즘의 경우 전체 오차가 적은 값을 갖더라도 일부 출력 뉴런은 상대적으로 높은 오차를 가진 상태에서 학습을 종료하는 경우가 있으므로, 소수의 학습패턴이 정확히 학습되지 못하고, 전체적으로 인식률을 저하시킬 수 있는 문제점을 가지고 있다.

따라서, 본 연구에서는 이러한 문제점을 해결하기 위하여 *Descending Epsilon* 방법을 이용하였다. Descending Epsilon 학습은 가중치를 조절하는 방법으로 기존의 오류 역전파 알고리즘과 동일하지만 학습과정을 수행하는 면에서 차이가 있다. 이 Descending Epsilon 학습방법의 주요 내용은 학습되지 않은 출력 뉴런만의 오차를 역전파하여 가중치를 조절하는 것으로 학습패턴의 오차의 크기가 학습 중에 허용할 수 있는 오차의 크기인 Epsilon보다 작으면, 그 학습패턴은 학습에 참여시키지 않는다. 따라서 학습이 잘 되지 않은 패턴에 대해서는 집중적인 학습을 수행할 수 있다는 장점을 지니고 있다. 따라서 신경망의 학습된 정도를 나타내는 척도로

$$E=\sum_{p=1}^{P}\sum_{i=1}^{N}(T_{ip}-O_{ip})^2 \qquad (5\text{-}8)$$

으로 계산되는 오차를 사용하는 대신

$$E=\sum_{p=1}^{P}\sum_{i=1}^{N}(Nmber\ of(\mid t_{ip}-O_{ip}\mid)>\varepsilon) \qquad (5\text{-}9)$$

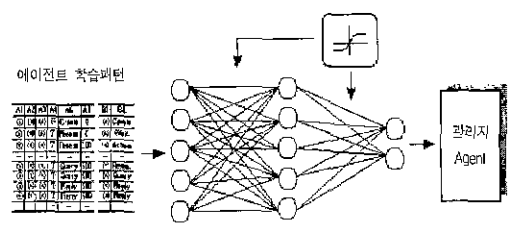
을 사용한다.

3.3 신경 스케줄링 모듈 (Neural Scheduling Module)

학습의 입력데이터는 사용자의 우선순위(A1), 현 작업을 수행중인 에이전트들의 이름(A2), 목적에이전트(A3)와 사용 가능 여부(A4), 목적에이전트에 의해 수행될 작업내용(A5), 시스템의 자원 사용량(A6)를 학습데이터로 사용하고, 수행될 작업에이전트 이름(B1) 및 작업내용(B2)을 교사데이터로 사용하여 총 7개의 필드로 구성된 레코드 단위를 구성하였다

신경회로망의 입력노드 값으로는 한 단위 시간씩 움직여가며 학습패턴을 구성하는 슬라이딩 윈도우 기법(Sliding Window Method)을 이용하며, 입출력시 사용될 데이터는 0~1 사이의 값으로 정규화 되어 계산된다.

초기 구성된 신경망은 6개의 입력노드와 5개의 은닉노드, 2개의 출력노드로 구성된 6×5×2의 신경회로망을 사용하며, 학습방법은 위에서 기술한 Descending Epsilon 학습방법을 이용하여 학습이 이루어지지 않은 패턴에 대한 집중적인 학습을 시도한다. 또한 각 층의 입력총합을 활성화함수에 적용시켜 각 노드의 출력값을 정한다. 이때 사용하는 활성화함수로는 양극성 시그모이드 함수를 사용하여 -1(음) 또는 1(양)의 값에 근사하게 되는 출력값을 갖도록 조정된다.



(그림 2) 신경스케줄링 모듈

4. 실험 및 고찰

본 논문에서 제안한 신경스케줄링 구축을 위하여 PentinumIII Processor하에서 Delphi를 이용하여 구축하였고, 각 에이전트는 JAVA 1.2.2를 이용하여 구현하였다[8, 11].

4.1 신경스케줄링을 위한 신경망 학습환경

신경 스케줄링 모듈 작성을 위하여 구성된 신경망 모델 구조는 입력노드 6개, 은닉노드 5개, 출력노드 2개로 구성된 6×5×2 구조로 구성하였다. 작성된 신경망 구조의 은닉노드수는 경험적 방식에 의하여 임의로 5개로 설정하였다. 이는 은닉노드의 개수가 많으면 학습의 속도가 느려지고 또한 학습이 제대로 수행되지 않는 경우가 발생하고 은닉노드의 개수가 매우 적으면 학습이 이루어지지 않는 상황이 발생하기 때문이다. 따라서 설계한 신경망 모델은 입력노드보다 1개가 작은 5개 노드로 임의 구성하였다.

입력노드와 은닉노드 사이의 활성화 함수는 양극성 시그모이드 함수를 사용하였고 각 입력층과 은닉층, 출력층간의 연결은 완전연결 되어 있다. 그리고 신경망 학습방법은 Descending Epsilon 학습 알고리즘을 사용하여 학습이 이루어지지 않은(미학습) 패턴에 대하여 집중적인 학습을 시도하였고 학습률은 0.4, 모멘텀은 0.7, 반복학습횟수는 200회로 설정하여 학습오차값이 지정된 오차값 이하로 감소하는 시점에 학습이 완료된 것으로 간주하였다.

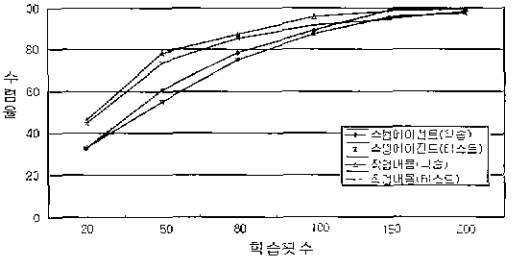
4.2 신경망 학습을 이용한 에이전트 신경스케줄링 모듈 작성

신경망 학습에 참여한 에이전트의 학습 패턴수는 사용자 우선순위(1~5), 현재 작업을 수행중인 에이전트, 작업수행을 요청할 목적 에이전트, 목적에이전트의 시용여부, 의뢰작업내용, 에이전트 사용가능량 등을 각각 곱한 입력패턴의 조합으로 구성된 총 8750건의 데이터 중 2000개를 임의 추출하여 학습데이터로 학습하였다.

학습은 예측된 목적 에이전트와 작업내용이 실제 패턴내용과 오차범위 이내이면 수렴된 것으로 간주하였고, 또한 지정된 학습횟수에 도달하였을 때에도 학습이 완료된 것으로 간주하였고, 학습 후 나머지 6750건의 데이터를 테스트패턴으로 이용하여 구성된 신경망의 적합도를 실험하였다.

〈표 2〉 학습패턴과 테스트패턴의 수렴결과

인식률 \ 학습횟수		20	50	80	100	150	200
B1 (수행 에이전트)	학습패턴	33.02	60.87	78.17	89.10	93.95	99.55
	테스트패턴	32.50	54.88	75.15	87.63	95.32	97.44
B2 (작업내용)	학습패턴	45.99	78.48	87.41	95.61	98.16	99.66
	테스트패턴	44.59	73.62	81.50	91.69	96.26	98.28



(그림 3) 수렴률 그래프

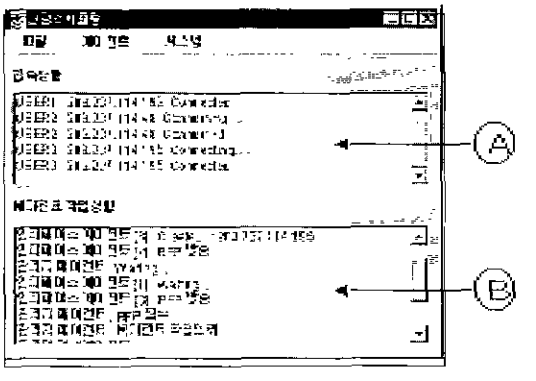
다음 <표 2>와 (그림 3)은 구성된 신경망(6×5×2)에서 학습을 통하여 구성된 신경망을 이용하여 학습패턴과 테스트 패턴의 수렴결과와 그래프를 나타낸다.

위의 <표 2>에서 두 개의 목표패턴인 수행에이전트(B1)와 작업내용(B2)은 학습횟수 200회에서 모두 99% 이상의 수렴률을 얻었다. 신경망 학습에 사용했던 학습패턴의 경우 학습횟수 100회에서 수행에이전트(B1)와 작업내용(B2)에 대하여 89.10%, 95.51%를 200회에서 각각 99.55%, 99.66%를 얻었다. 그러나 학습이 이루어지지 않은(비 학습)데이터인 테스트패턴의 경우 학습패턴에 비하여 수렴률이 약간 적게 나왔지만 수행에이전트(B1)와 작업내용(B2) 모두 학습횟수 200회에 대하여 97.44%, 98.28%를 얻어 학습데이터와 거의 동일한 수렴률을 얻었다. 이는 신경망이 가지고 있는 일반성의 능력으로 비 학습패턴에 대해서도 학습패턴의 내용을 적용하여 예측 및 분석이 가능함을 보여준다. 실험을 통하여 얻어진 신경망 학습데이터는 멀티에이전트 시스템의 신경스케줄링 데이터로 활용이 된다.

4.3 신경스케줄링을 적용한 시뮬레이션

위에서 실험한 신경망 학습결과를 시스템 관리자 에이전트 신경모듈에 입력하여 각 에이전트에 대한 시스템조건과 에이전트상태 그리고 사용자의 우선순위에 대한 그 적용여부를 실험하였다.

아래의 (그림 4)는 구현된 관리자 에이전트 모듈이 실행되고 있는 과정으로 작업수행을 위하여 접속한 사용자의 현황과 각 에이전트들의 작업상황을 나타낸다. 또한 관리자 에이전트는 작업 요청을 위해 접속한 사용자를 위하여 해당 에이전트를 생성하고, 이에 맞는 작업을 지시하는 일을 수행하고 있다.



(그림 4) 작업중인 관리자 에이전트

위의 그림의 Ⓐ영역은 시스템에 접속한 사용자의 접속현황 및 접속위치(IP)를 보여주고, Ⓑ영역은 사용자 접속상황에 따른 에이전트의 생성 및 작업수행을 위하여 각 에이전트들이 작업의 내용을 분할하여 상호 협력하는 과정들을 나타내고 있다. 아래의 (그림 5)는 Ⓑ 영역의 일부내용이다

```
인터페이스 에이전트 [1] . Create . : 203.237 114.193
인터페이스 에이전트 [2] : Create . . 203.237 114.48
인터페이스 에이전트 [3] : Create . 203.237 114.195
인터페이스 에이전트 [2] : RFP 발송
관리자 에이전트 : RFP 접수
인터페이스 에이전트 [1] . Waiting .
인터페이스 에이전트 [3] : RFP 발송
관리자 에이전트 . RFP 접수
관리자 에이전트 : 에이전트 작업의뢰 .
질의어 검색에이전트[1] : Create.
질의어 검색에이전트[2] . Create.
인터페이스 에이전트[1] : RFP 발송
관리자 에이전트 : RFP 접수
```

(그림 5) 에이전트의 작업상황

위의 (그림 5)는 사용자 접속위치(IP)에서 사용자가 접속하게 되면 관리자 에이전트는 인터페이스 에이전트를 생성하여 사용자의 접속요구를 받게 된다. 위의 그림에서는 세 명의 사용자가 동시에 접속을 시도하였을 때 각각 세 개의 인터페이스 에이전트를 생성하고 작업요청을 받고 있다. 이후 [2]번 인터페이스 에이전트가 사용자로부터 질의어를 입력받아 이에 대한 RFP를 작성하여 관리자 에이전트에게 발송하고, 관리자 에이전트는 RFP를 접수한 후 질의어 에이전트에게 작업의뢰를 시행하는 과정을 나타내고 있다. 실험을 통하여 관리자에이전트에 입력된 신경망 학습결과가 에이전트의 생성, 작업수행, 분할 및 상호협력과정을 잘 수행함을 보여주고 있다.

5. 결 론

분산문제의 해결을 위한 에이전트 시스템은 에이전트간의 상호협력과 협동을 통하여 주어진 문제의 해를 찾아가는 시스템이다. 이러한 에이전트 시스템은 개념적으로나 물리적으로 매우 복잡하고 에이전트간의 상호작용이 많아 복잡해지지만 인공사회에서 필요성이 높은 응용분야에 적용할 수 있는 접근방식이다.

본 논문에서는 이러한 문제들의 해결방법으로 멀티 에이전트 시스템의 에이전트간 효율적인 작업연계를 위한 신경스케쥴링 방법을 제시하였다.

신경스케쥴링(Neural Scheduling) 작성을 위하여 시스템 사용에 대한 시나리오를 바탕으로 사용자의 우선순위, 현 작업을 수행중인 에이전트이름, 목적에이전트와 사용 가능여부, 목적에이전트에 의해 수행될 작업에이전트의 이름, 시스템 자원의 사용가능량을 학습데이터로 사용하고 수행될 작업에이전트의 이름 및 작업내용을 교사데이터로 사용하여 총 7개의 필드로 구성된 레코드단위의 총 8750건의 패턴을 구성하였고 이중 2000건을 학습데이터로, 나머지 6750건을 테스트패턴으로 사용하여 6×5×2의 신경망 구조에서 학습하였다 학습결과 학습횟수 100회에서 약 90%정도의 수렴율을 보였고, 학습회수 200회에서 학습패턴과 테스트패턴에 대하여 모두 99%이상의 수렴율을 얻을 수 있었다

또한 학습된 결과를 관리자 에이전트(MA)의 신경모듈에 입력하고 각 에이전트에 대한 적용여부를 실험하여 에이전트의 생성, 작업의 수행, 분할 및 상호협력과정을 잘 수행함을 보였다.

향후 연구과제로 다중 에이전트 시스템상에서 신경스케쥴링을 작성하기 위한 주요 요인들에 대한 연구와 함께 시스템이나 에이전트가 추가되었을 때 신경스케쥴링을 재구축하여야 하는등의 문제점을 해결할 수 있는 연구가 수행되어져야 할 것으로 사료된다

참 고 문 헌

[1] John. K Black, et al., "Real-time Method Invocations in Distributed Environments," University of Rhode Island Department of Computer Science and Statics, Technical Reports TR95-244, 1995.

[2] "Scheduling in Real-Time Systems" http://www.realtime-os.com/sched_03.html. 1996

[3] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science, Vol.220, pp.671-674, 1983.

[4] L. J. Park and C. H. Park, "Genetic algorithms for job shop scheduling problems based on two representational schemes," Electronic Letter, Vol.31, No.23, pp.2051-2053, 1995.

[5] J. Carlier and E. Pinson, "An algorithm for solving the job shop problem," Management Science, Vol. 35, pp.164-176, 1989

[6] Y. H. Yu, R F. Simmons, "Descending Epsilon in Back-Propagation : A Technique for Better Generalization," IEEE IJCNN, Vol.3. pp.167-172, 1990.

[7] S. Franklin, A. Graesser, "Is it an Agent, or just

a Problem? : A Taxonomy for Autonomous Agents," University of Memphis, 1996
[8] David Flanagan, "Java in a Nutshell, A Desktop Quick Referecne for Java Programmers." O'Reilly & Associates, Inc, 1996.
[9] S. Russel and P.Norvig, Artificial Intelligence a mordern approach, Prentice Hall, 1995
[10] A Baratloo, M. Karaul, H. Karl, Z. M. Kedem, KinttingFactory : An infrastructure for Distributed Web Applications, New York Univ. Nov 13, 1997
[11] B. O Christiansen, P. Cappello, M, F. Ionescu, M. O. Neary, K E. schauser, D. Wu, "Javelin : Internet-Based parallel Computing Using Java," *ACM Workshop on java for science and Engineering compution*, (jun 1997).

이 기 준

e-mail : cholee@shinbiro.com
1994년 조선대학교 전산통계학과 (이학사)
1997년 조선대학교 일반대학원 전산통계학과(이학석사)
1998년~현재 조선대학교 일반대학원 전산통계학과 박사과정
관심분야 : 신경망, 패턴인식, 인공지능, 분산 에이전트 시스템

정 종 필

e-mail : jp5678@scjc.ac.kr
1992년 조선대학교 전산통계학과 졸업(이학사)
1995년 조선대학교 대학원 전산통계학과 졸업(이학석사)
1998년 조선대학교 대학원 전산통계학과 박사과정수료
1997년~현재 순천청암대학 컴퓨터정보과학부 전임강사
관심분야 : 영상처리, 신경망, 멀티미디어, 컴퓨터 애니메이션

정 채 영

e-mail : cyjung@mail.chosun.ac.kr
1983년 조선대학교 컴퓨터공학과 (이학사)
1986년 조선대학교 일반대학원 전자과 전산전공(공학석사)
1989년 조선대학교 일반대학원 전기과 전산전공(공학박사)
1986년~현재 조선대학교 자연과학대학 수학·전산통계학부 부교수
관심분야 : 영상처리, 신경망, 데이터베이스, 멀티미디어 컨텐츠