한국게임학회 논문지
제7권 제4호 2007년 12월

조정 에이전트를 이용한 작업 할당 최적화 기법

박재현[0], 조경은*, 엄기현**
(주)리호커뮤니케이션[0], 동국대학교 영상미디어대학 게임멀티미디어공학과*,**
{jhyun[0], cke*, khum**}@dongguk.edu

An Optimization Strategy of Task Allocation using Coordination Agent

Jaehyun Park[0], Kyungeun Cho*, Kyhyun Um**
Liho Communication Ltd.[0], Dept. of Game Multimedia Engineering, Dongguk University*,**

요 약

게임과 같은 실시간이며 복잡한 다중 에이전트 환경에서는 시스템의 효율성을 극대화하기 위해 반복적으로 작업 할당이 수행된다. 본 논문에서는 실시간 다중 에이전트 구조에 적합하며 최적화된 작업 할당이 가능한 방안으로 A* 알고리즘을 적용한 조정 에이전트를 제안한다. 제안하는 조정 에이전트는 수행 가능한 에이전트와 할당 가능한 작업으로 정제된 모든 에이전트와 작업의 조합으로 상태 그래프를 생성하고, A* 알고리즘을 이용한 평가함수를 적용하여 최적화된 작업 할당을 수행한다. 또한 실시간 재할당에 따른 지연을 방지하기 위해 그리디 방식을 선택적으로 사용함으로써 재할당 요구에 대한 빠른 처리가 가능하다. 마지막으로 모의실험을 통해 조정 에이전트를 통한 최적화된 작업 할당 결과가 그리디 방식의 작업 할당보다 성능이 25% 향상되었음을 입증한다.

ABSTRACT

In the complex real-time multi-agent system such as game environment, dynamic task allocations are repeatedly performed to achieve a goal in terms of system efficiency. In this research, we present a task allocation scheme suitable for the real-time multi-agent environment. The scheme is to optimize the task allocation by complementing existing coordination agent with A* algorithm. The coordination agent creates a status graph that consists of nodes which represent the combinations of tasks and agents, and refines the graph to remove nodes of non-execution tasks and agents. The coordination agent performs the selective utilization of the A* algorithm method and the greedy method for real-time re-allocation. Then it finds some paths of the minimum cost as optimized results by using A* algorithm. Our experiments show that the coordination agent with A* algorithm improves a task allocation efficiency about 25% highly than the coordination agent only with greedy algorithm.

Keyword : Multi-Agent, Task Allocation, Coordination Agent, A* Algorithm

※교신저자(Corresponding Author):조경은, 주소:서울시 중구 필동 3가 26 동국대학교(100-715), 전화:02)2260-3834, E-mail:cke@dongguk.edu

1. 서론

실시간이며 광범위한 영역에 자원과 에이전트가 분산되어 있는 현실 세계의 문제를 해결하기 위해서는 자원과 구조가 제한적인 단일 에이전트 구조로는 한계가 있다. 이러한 복잡한 문제를 실시간으로 해결하기 위해서는 분산된 자원을 기반으로 자체 구조가 능동적으로 변하는 다중 에이전트 시스템이 적합하다 [1]. 다중 에이전트 시스템은 에이전트간의 합리적인 상호협동으로 전체 균형을 유지하면서 목표 달성이 가능하며, 현재 네트워크 제어 및 관리, 통신, 스위칭, 서비스 관리, 전자 상거래, 게임 등 다양한 분야에 적용되고 있다.

실시간 환경에서 다중 에이전트 시스템은 작업 할당문제 해결을 동적으로 처리하기 때문에 동적 작업 할당에 대한 연구가 필요하다. 작업 할당이 동적으로 이루어지기 위해서는 에이전트간의 의존성과 작업 수행에 따른 충돌을 고려한 할당과 불확실한 환경에 대해 능동적 처리가 가능하고 유연한 시스템 구조 설계가 필요하다. 또한 작업 할당은 효율성을 고려한 최적의 결과를 이끌어내야 한다. 따라서 최적의 결과를 보장하는 작업 할당 최적화 방법은 실시간 문제 해결을 위해 해결해야할 중요한 과제다.

동적으로 작업을 할당하고 배정하는 방법은 조정 에이전트와 같은 관리자 역할의 에이전트를 통한 중앙 집중적인 방식과 에이전트간 자동화된 계약, 협상을 통한 분권적인 작업 할당 방식이 있다 [2]. 중앙 집중적인 방식의 대표적인 작업 할당 방법으로는 단순하면서도 빠른 할당이 가능한 그리디(greedy) 알고리즘 방식이 사용된다 [3]. 그러나 상황이 급변하는 실시간 환경에서 빠른 할당에는 유용하지만, 할당 결정에 대한 예측이 불안정하며 최적의 결과를 보장하지 않는다. 반면 분권적인 방식은 모든 에이전트가 작업 할당 의사결정에 참여하는 형식으로 합의를 통한 최적의 결과가 가능하다 [4]. 분권적인 방식으로는 대표적으로 경매방식이 있다.

본 논문에서는 실시간이며 복잡한 다수의 에이전트가 존재하는 게임 환경과 같은 다중 에이전트 구조에 적합하며 최적화된 작업 할당이 가능한 방안으로 기존의 중앙 집중적 작업 할당 방식을 보완하는 방법을 제안한다. 작업을 할당하는 역할의 조정 에이전트는 A* 알고리즘을 적용하여 최적의 결과가 가능하도록 한다. A* 알고리즘을 통한 작업 할당은 각각의 에이전트와 할당 가능한 모든 작업의 조합을 하나의 상태로 보고 최적의 결과를 탐색하는 방식을 사용한다. 그러나 탐색 영역의 증가에 따른 할당 지연을 방지하기 위해 현재 수행 가능한 작업과 에이전트만을 선별하는 필터링 역할의 정제 기능을 추가 보완한다. 또한 실시간 환경에 적합한 유동적인 탐색 방식 선택으로 환경의 불확실성에 대한 시스템의 견고성을 높이도록 한다. 따라서 실시간 환경에 적합한 중앙 집중적 방식의 한계인 최적화 보장 문제를 보완하여 실시간 다중 에이전트 시스템의 성능 향상 및 작업의 효율성 증가를 연구 목적으로 한다.

2. 관련 연구

2.1 다중 에이전트 시스템의 조정 방법

다중 에이전트 시스템에서는 에이전트들이 효율적으로 상호 협동하면서 공통의 목표가 완성되는 특징을 가지고 있다. 그러나 작업 및 자원의 충돌은 에이전트간의 제한적인 정보교환과 한정된 자원으로 인해 에이전트가 중복된 작업을 수행하거나 공통의 자원을 점유하기 위해 대기하는 문제가 발생할 수 있다. 또한 다중 에이전트 시스템은 작업을 수행하기 위한 모든 정보와 자원에 대한 상호 의존성이 존재한다. 따라서 에이전트간 충돌 발생요인을 제거하고, 분산된 자원과 정보를 통합 관리하며, 상호 의존성을 고려한 합리적인 작업 분배를 위해서는 에이전트와 작업에 대한 조정(coordination) 기술이 필요하다 [1]. 다중 에이전트 시스템의 문제점을 해결하기 위한 조정 기술은 작업의 효율성을 증가시켜 최적화된 결과를 이끌어낸다. 다중 에이전트 시스템 연구에서 조정의 의미는 협소한 의미와 포괄적인 의미 두 가지로 나눌 수 있다. 협소한 의미의 조정은 계획 전후 충돌을 방지하기 위한 처리 과정을 의미한다 [5]. 반면 포괄적인 의미의 조정은 목표를 분할하고 작업을 할당 및 배정하는 전반적인 과정을 의미한다 [6].

2.2 다중 에이전트 시스템의 작업 할당 접근 방식

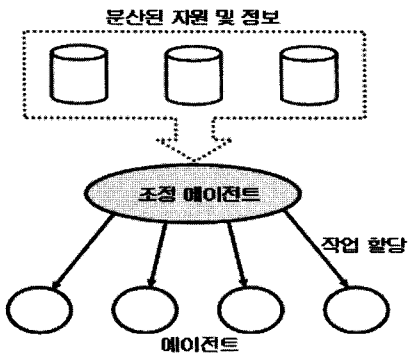
작업 할당은 실행 가능한 작업들을 각 에이전트의 능력과 역할에 따라 배정하여 효율적으로 작업을 수행하도록 하는 것을 의미한다. 작업의 단위 구성은 장기적인 관점의 전역적인 목표를 수행하기 위한 복합작업(composite task)과 이

러한 복합 작업들을 구성하고 있는 단기적 목표인 구성작업(component task)들로 이루어진다. 또한 구성작업은 하나의 에이전트가 수행하는 행동들의 집합인 행위(behavior)들로 이루어진다 [7]. 본 논문에서 사용하는 일반적인 작업의 의미는 에이전트가 수행할 수 있는 구성작업을 의미한다.

작업 할당을 수행하는 방법에는 작업 할당의 권한이 중앙에서 처리되는 중앙 집중적인 방법과 분산된 형태의 분권적인 방법이 있다 [2].

2.2.1 중앙 집중적인 작업 할당

중앙 집중적인 작업 할당에서는 관리자 역할을 담당하는 조정 에이전트를 사용하여 다중 에이전트에 대한 계획을 통합 관리한다. 조정 에이전트는 [그림 1]과 같이 서로 다른 고유의 능력을 가지고 있는 에이전트에게 전역적인 목표를 완성하기 위한 작업들을 할당하는 역할을 담당하고 있다. 따라서 사용할 수 있는 모든 자원과 행동에 대한 정보를 가지고 있어야 하며, 각각의 에이전트가 계획을 수행함에 있어 자원에 대한 충돌이나 중복된 계획이 없도록 조정해 주는 역할을 수행한다.

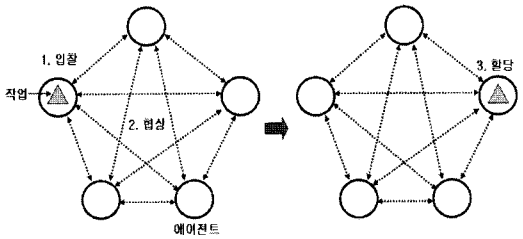


[그림 1] 중앙 집중적 작업 할당

결과적으로 에이전트는 조정 에이전트로부터 할당 받은 작업을 수행함으로써 전역적인 목표가 완성된다. 대표적인 중앙 집중적인 작업 할당 방법으로는 단순하면서도 최적화가 가능한 그리디 알고리즘을 이용한 방식이 사용된다 [1]. 그리디 방식의 경우 일련의 선택 단계마다 최선의 선택을 해나가는 방식으로 최상의 유효 조건을 반복적으로 검색하면서 최적 할당이 이루어진다. 그러나 각 단계의 최적이 전체의 최적을 보장하지 않기 때문에 작업 할당 최적화가 보장되지 않는 문제점이 있다 [8].

2.2.2 분권적인 작업 할당

분권적인 작업 할당 방법은 [그림 2]처럼 모든 에이전트가 동등한 입장에서 의사 결정에 참여하는 방식으로 에이전트 간의 상호 계약, 협상 합의를 통한 최적 할당이 가능하다. 대표적인 분권적인 작업 할당 방법으로는 경매 방식이 있으며, 지금까지 에이전트간의 자동화된 합의를 통한 효율적인 작업 할당을 위해 다양한 경매 방식이 연구되어 왔다. 경매 방식을 적용하면 구현이 용이하고 작업 할당 과정이 협상 합의를 통해 효율적으로 처리되며 에이전트들이 충분한 영역지식을 갖지 못한 경우에도 문제 해결이 가능한 장점이 있다. 그러나 경매방식은 협상을 통한 합의에 도달하기 위해 반복된 협상을 수행하고 상호 작용에 대한 규정인 공통의 협상 프로토콜을 사용한다. 따라서 의사결정에 많은 시간이 소요되며 에이전트간의 통신으로 인한 오버헤드가 발생한다. 따라서 다양한 경매 방식을 적용하기 위해서는 에이전트들의 상호작용에 대한 규정이 되는 공통의 협상 프로토콜이 필요하다.



[그림 2] 분권적인 작업 할당 방식

2.3 실시간 다중 에이전트 시스템의 작업 할당 최적화

다중 에이전트 시스템의 작업 할당을 최적화하기 위한 방법은 다양한 접근 방식으로 여러 가지 방법이 제안되었다. 그러나 각각의 방식은 상대적인 장단점이 존재하기 때문에 현재까지는 일반화된 최적의 방법이 없다.

구분	방 식	장 점	단 점
할당	중앙집중식	빠른 할당	최적화 문제
방식	분 권 적	최적 할당	수행 지연

[표2] 최적화 방식에 따른 장단점

분권적인 작업 할당 방법의 보완 방안으로 협상 시간을 단축하기 위한 사전정의작업과 중재에이전트를 두는 방법이 있다. 그러나 사전정의작업으로 에이전트간의 정보를 제한하고 협상시간에 대한 시간한계를 적용하면 한정된 정보를 가지고 한정된 시간에만 할당이 이루어지기 때문에 제한적인 작업 할당이 이루어진다. 또한 기존의 중재 에이전트는 할당을 최적화로 이끌어가는 역할이 아니라 단순히 에이전트의 정보를 수집해서 알려주는 커뮤니케이션 서버의 기능을 수행하고 있기 때문에 결과적으로 최적의 할당은 에이전트간의 협상으로 이루어진다.

중앙 집중적인 작업 할당 방법은 작업을 수행하기 적합한 에이전트를 탐색하고 작업을 할당하는 모든 과정이 조정 에이전트를 통해 처리되는 방식이기 때문에 분권적인 방식보다 적은 비용과 시간으로 할당이 가능하다. 따라서 할당 결정에 대한 최적화 문제점을 보완한다면 중앙 집중적인 방식이 실시간 동적인 환경에 적합하다 [9].

3. A* 알고리즘을 이용한 작업 할당

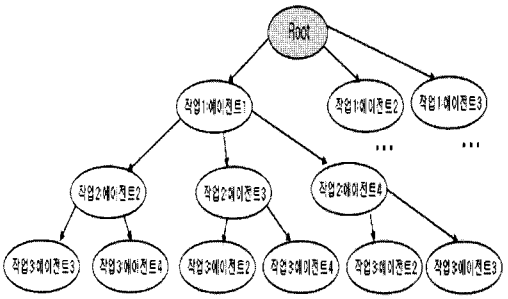
본 논문에서는 실시간 다중 에이전트 시스템에서 기존의 중앙 집중적 방식인 그리디 알고리즘을 적용한 조정 에이전트를 보완하여 동적 작업 할당 및 최적화가 가능한 방안을 제안한다. 최적화 방법으로는 휴리스틱 값을 이용한 최단 거리 탐색 방법인 A* 알고리즘을 이용한다 [11]. 작업 할당을 A* 알고리즘을 이용하여 수행함으로써 탐색 공간에서 최적화 조건을 만족하는 에이전트에게 작업이 할당된다. 제안하는 조정 에이전트는 기존의 그리디 방식에 비해 최적화 성능을 향상시키는 것을 목적으로 한다.

3.1 동적 작업 할당 방안

작업은 다중 에이전트 시스템의 궁극적인 목표를 수행하기 위해 에이전트가 수행할 수 있는 여러 가지 행동이나 역할로 조정 에이전트를 통해 할당된다. 조정 에이전트는 작업과 에이전트의 상태 조합에서 그리디 방식과 A* 알고리즘 방식 두 가지 방식을 동적으로 적용함으로써 실시간 처리가 가능하다.

3.1.1 작업 할당 기본 개념

A* 알고리즘은 지향성(directed) 알고리즘으로 가장 적절한 탐색 방향을 평가하며 목표지향적인 방식으로 길을 찾는다. 일반적으로 A* 알고리즘은 두 지점간의 경로가 여러 개가 존재할 때 최단 경로를 찾는 알고리즘으로 사용된다. 본 논문에서 A* 알고리즘은 [그림 3]과 같이 각각의 에이전트와 할당 가능한 모든 작업의 조합으로 이루어진 상태 조합 그래프에서 최적화된 작업 할당을 탐색하는 알고리즘으로 사용한다.



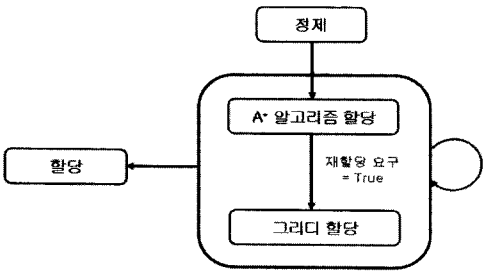
[그림 3] 작업과 에이전트의 상태 조합 그래프

상태 조합 그래프는 n개의 에이전트 집합 $A = \{a_1, a_2, \ldots, a_n\}$와 r개의 작업으로 구성된 집합 $T = \{t_1, t_2, \ldots, t_r\}$에서, n개의 에이전트가 r개의 작업을 할당받아 수행할 수 있는 모든 가능한 방법의 수인 중복조합 nHr으로 표현가능하다. 따라서 에이전트와 작업의 조합으로 구성된 집합 $U = \{ (a_1, t_1), (a_2, t_2), \ldots, (a_n, t_r) \}$는 A* 알고리즘으로 탐색을 적용하게 될 탐색 공간의 각 노드가 된다.

그러나 조정 에이전트가 A* 알고리즘으로만 작업 할당이 수행될 경우 탐색 공간의 증가로 인한 할당 지연이 발생하기 때문에 실시간 처리 문제가 발생한다. 따라서 제안하는 조정 에이전트에서는 A* 알고리즘에서 탐색에 의한 할당 지연을 방지하기 위해 탐색이 이루어지기 전에 에이전트와 작업에 대한 정제(refinement)가 수행된다. 정제는 현재 수행할 수 없는 작업과 작업을 수행할 수 없는 에이전트를 제거하는 기능을 가지고 있다. 이는 탐색 영역을 감소시킴으로써 할당 지연을 방지한다.

실시간으로 에이전트 및 자원에 대한 정보가 변하는 동적인 환경에서는 작업 할당을 위해 A* 알고리즘으로 탐색 진행 중, 새로운 환경에 대한 작업의 재할당을 요구하는 일이 발생한다. 이때 [그림 4]와 같이 조정 에이전트는 현재 진행

중인 모든 작업을 중단하고 그리디 방식으로 작업을 할당함으로써 재할당 요구에 대한 빠른 처리가 가능하다.



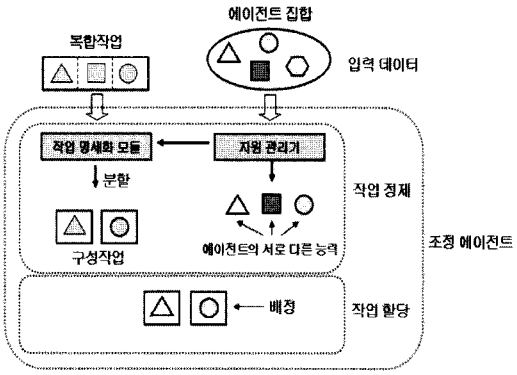
[그림 4] 조정 에이전트를 이용한 동적 작업 할당

제안하는 조정 에이전트는 A* 알고리즘을 이용한 평가 함수를 적용하여 작업 할당이 수행되기 때문에 최적화된 결과가 보장되며, 유동적으로 작업 할당 알고리즘을 적용하여 지연에 대한 문제점을 보완하기 때문에 실시간 다중 에이전트 환경에 적합한 작업 할당을 위한 조정 에이전트 구현이 가능하다.

3.1.2 조정 에이전트의 역할

조정 에이전트의 역할을 좀 더 세부적으로 살펴보면 [그림 5]와 같이 전역적인 목표를 수행하기 위한 복합작업을 각각의 에이전트가 수행할 수 있는 단기적 목표인 구성작업으로 나누고, 에이전트의 능력에 맞게 목표를 배정 및 할당하는 것이다. [그림 5]에서 에이전트의 다각형은 에이전트가 가지고 있는 고유의 역할 및 능력을 의미하며, 작업의 다각형은 작업을 수행하기에 적합한 역할 및 능력을 의미한다. 따라서 작업 모양에 맞는 에이전트를 할당하는 것이 최적의 의미를 나타낸다.

조정 에이전트의 세부적인 기능 및 정의는 다음과 같다. 복합작업은 에이전트가 수행할 수 있는 다수의 구성작업으로 이루어졌으며, 이러한 구성작업들의 수행 결과로 복합작업이 완료된다. 복합작업의 분할은 시스템 설계시 정의된 작업 명세서에 따라 이루어진다. 에이전트는 특정 목적에 대한 작업을 수행하는 자율적인 프로세스이며 고유의 역할 및 특징을 가지고 있다. 작업 및 에이전트의 정제는 조정 에이전트의 주요기능으로 자원관리기를 통해 현재 자원의 상태와 에이전트 정보를 분석하여, 수행할 수 없는 작업과 작업을 수행할 수 없는 에이전트를 제거하는 단계를 의미한다. 마지막으로 할당은 작업을 수행하기에 적합한 에이전트를 탐색하여 작업을 배정한다. 탐색은 [그림 3]과 같은 상태 그래프에서 A* 알고리즘의 평가함수를 이용하여 최단경로를 찾는 방식과 동일한 방법으로 수행된다. 제안하는 조정에이전트에서는 A* 알고리즘을 적용한 방식과 그리디 방식을 동적으로 사용하기 때문에 실시간 환경에 적합한 구현이 가능하다.

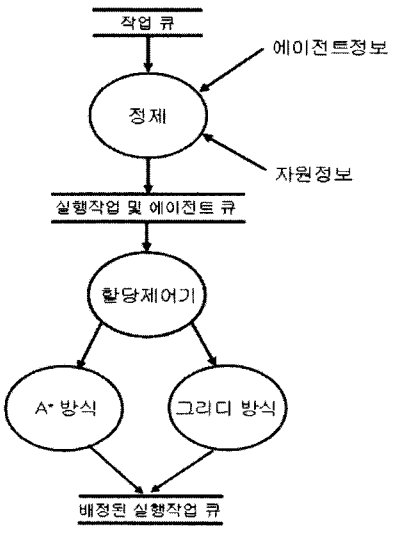


[그림 5] 조정 에이전트의 기능

3.1.3 조정 에이전트의 구조

(1) 조정 에이전트의 설계 및 구현

제안하는 조정 에이전트는 복합작업 및 구성작업과 에이전트가 저장된 데이터 자료구조인 큐, 그리고 정제와 할당의 주요기능을 담당하는 두 개의 클래스로 구성된다. 조정 에이전트를 DFD로 설명하면 [그림 6]과 같다.



[그림 6] 조정 에이전트 DFD

작업큐는 복합작업을 분할한 구성작업들이 저장된 자료구조이며 모든 에이전트와 시스템 자원에 대한 정보를 이용하여 정제를 수행한다. 정제된 작업과 에이전트는 실행작업 및 에이전트큐에 저장된다. 이는 할당을 수행하기 위한 입력 데이터로 사용된다. 실행작업 및 에이전트큐는 정제된 작업과 이를 수행할수 있는 정제된 에이전트들이 저장된 자료구조이며, 실행작업 및 에이전트큐에 저장된 데이터를 기반으로 할당제어기는 최적의 탐색을 수행하기 위한 작업과 에이전트의 조합으로 구성된 트리형태의 검색 공간을 생성하게 된다. 탐색은 작업과 에이전트의 수행시간 및 예상시간을 기준으로 평가 함수를 적용한 후, 탐색공간에서 최소비용이 가능한 조합을 선택하는 A* 알고리즘을 적용하여 이루어진다.

(2) A* 알고리즘 방식

A* 알고리즘에서는 노드 탐색 데이터와 전통적인 '열린', '닫힌' 노드들을 담을 자료구조가 필요하며, 어떤 자료구조를 이용할 것인지에 대한 문제는 A* 알고리즘 처리 성능에 영향을 준다. 또한 '임의의 개수' 의 자식들을 가질 수 있으려면 확장성이 좋은 자료구조가 필요하다. 따라서 각각의 작업과 에이전트의 조합을 트리 구조로 구성한다. [그림 7]은 할당에 적용한 A* 알고리즘으로 부모 노드가 비용과 휴리스틱 값에 따라 자식 노드를 생성하면서 트리가 구성됨을 알 수 있다. 탐색은 '열린' 노드가 남아있는 동안 반복적으로 수행되며, '열린' 노드 가운데 최소인 노드를 '닫힌' 노드에 넣게 된다. 만일 동일한 노드가 여러 개 있을 경우에는 임의로 선택하되 목표노드가 있다면 우선적으로 선택된다.

A* 알고리즘이 최적으로 작동하게 하는 핵심은 비용과 휴리스틱 값을 계산하는 함수에 있다. 비용과 휴리스틱 값은 적용하는 상황에 따라 달라질 수 있으며 A* 알고리즘을 적용하여 선택 가능한 경로들 가운데 결정의 기준이 되거나 영향을 주는 요인을 비용으로 정의한다 [10]. 예를 들면 길찾기 상황에서는 두 지점간의 거리나 이동 시간 또는 소요비용 등을 비용으로 정의하고 평가함수에 적용함으로써 최단거리, 최소시간, 최소비용을 결정하는 기준이 된다. 또한 휴리스틱 값은 탐색되는 노드의 적합성과 A* 알고리즘을 평가하는데 쓰인다.

(3) 그리디 방식

그리디 방식은 단계별 최대 이익이 결과적으로 전체적인 최대 이익이라는 최적성의 원리에 따라 작업을 할당한다. 따라서 작업과 에이전트가 실시간으로 추가되거나 소멸되는 각 단계마다 최상의 조건을 반복적으로 선택하면서 할당이 수행된다. 그리디 알고리즘의 할당 방식은 [그림 8]과 같이 현재 시점에서 가장 우선순위가 높은 작업을 우선순위가 가장 높은 에이전트에게 할당한다.

실시간 환경에서는 임의의 시점에서 작업과 에이전트에 대한 할당을 수행하기 위한 연산 및 에이전트의 탐색이 진행중인 상황에도 재할당이 필요한 경우가 발생할 수 있기 때문에 A* 알고리즘으로 탐색이 진행 중인 상황에서도 재할당 요구에 따른 빠른 할당이 필요하다. 할당제어기는 이러한 상황을 처리하기 위해 재할당 요구가 있을시, A* 알고리즘으로 진행 중인 모든 작업을 중단하고 그리디 방식으로 할당한다. 따라서 재할당에 대한 빠른 할당 처리가 가능하며 재할당 요구를 충족시킨다.

```
Fantion : Search_by_Astar

Create Open_List and Closed_List
Put the Starting State on the Open_List

while(the Open_List is Not Empty)
{
    Find the Node with the Least F(n) on the Open_List, Call it Current
    Pop Current off the Open_List
    Generate all Legal States 1 State away from Current, Call them Children
    Set Current as Parent of all Children
    for(all Children)
    {
        Children[i].g = evaluateCost() ; the actual task execution time
        Children[i].h = evaluateExpectation() ; the estimated task execution time
        Children[i].f = Children[i].h + Children[i].g

        if(there is a State on the Open_List Or Closed_List that has the same
                position as Children[i] and a lower f(n) value)
        Skip this State
        else
        Put Children[i] on the Open_List
    }
    Put Current on the Closed_List
}
```

[그림 7] 할당에 적용한 A* 알고리즘

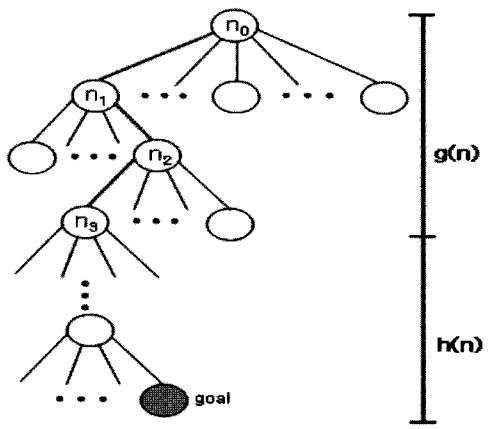
```
Function : Search_By_Greedy

for ( i=0 ; i < GoalCount ; i++ ){
    for ( j=0 ; j < AgentCount ; j++ ){
        if (!CheckAgent(i,j)){
         nPriority = j;
         break;
        }
    }
    for ( j=0 ; j < AgentCount ; j++ ){
        if (!CheckAgent(i,j))
          if ( EvaluatePriority(i,nPriority) < EvaluatePriority(i,j) )
           nPriority = j;
    }
}
```

[그림 8] 할당에 적용한 그리디 알고리즘

3.2 최적화 방안

3.2.1 조정 에이전트의 휴리스틱 평가함수

그래프 탐색에서 A* 알고리즘은 휴리스틱 평가함수를 사용하여 탐색한다. [그림 9]에서와 같이 트리에서 임의의 노드 n_3에 대하여 노드 n_3과 목표 노드 사이의 최단 경로 값을 $h(n)$으로 설정하고 $g(n)$은 시작 노드 n_0에서부터 n_3까지의 최단 경로 값으로 한다. 따라서 A* 알고리즘의 평가 함수는 [수식 1]로 표현하며 이는 n_0에서 시작하여 노드 n_3를 통하여 목표 노드까지 갈 수 있는 모든 가능한 경로 중 최단 경로의 값이 된다.



[그림 9] 그래프 구조의 탐색 공간

$$f(n)=g(n)+h(n) \qquad \text{[수식 1]}$$

본 논문에서는 최적화 평가함수로 상태 n에서 목표 노드까지의 경로 에 해당하는 값 $h(n)$을 할당된 작업을 수행하는 작업 수행 예상시간으로 설정하고 $h'(n)$으로 한다. 시작 노드로부터 경로에 해당하는 값 $g(n)$은 할당된 작업을 에이전트가 수행하기 위해 소요된 실제 작업 수행시간으로 설정하고 $g'(n)$으로 한다. 따라서 최적화 평가함수는 [수식 2]가 된다.

$$f'(n)=g'(n)+h'(n) \qquad \text{[수식 2]}$$

따라서 최적화 평가함수 [수식 2]를 이용하여 에이전트와 작업의 조합으로 이루어진 상태에 대한 탐색이 수행될 경우 최단 수행 시간이 예상되는 최적의 결과를 얻을 수 있다. A* 알고리즘이 최적의 결과를 보장하기 위해서는 휴리스틱 값에 대한 예측이 실제 측정값보다 작은 값이어야 한다는 조건을 만족해야 한다 [11]. [수식 2]에 적용한 휴리스틱에 대한 추정 요소는 작업을 수행하기 위한 에이전트 수에 대한 조건을 배제한 동시 작업 수행시간으로 예측하기 때문에, 실제 작업을 수행하기 위한 에이전트의 제한이 있을 경우보다 작은 값을 보장한다. 따라서 추정값 $h'(n)$은 실제값 $h^*(n)$보다 같거나 작아야 한다는 A* 알고리즘의 최적화 증명 조건 [수식 3]을 만족하며, 최적화 평가함수 [수식 2]는 최적의 결과를 보장한다.

$$h'(n) \leq h^*(n) \qquad \text{[수식 3]}$$

마지막으로 조정 에이전트에서 그리디 방식으로만 작업을 할당했을 경우는 평가함수 [수식 4]가 된다.

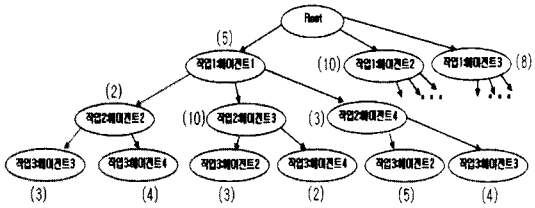
$$f'(n) = h'(n) \qquad \text{[수식 4]}$$

3.2.2 평가함수 적용 사례

(1) 그리디 방식

그리디 방식은 일련의 선택 단계에서 최선의 선택을 하는 방식으로 탐색이 수행된다. 따라서 [그림 10]과 같이 시작노드에서는 $h'(n)$ 값이 가장 작은 {작업1:에이전트1}이 선택되며, 다음 선택을 하기 위해 이웃한 노드의 $h'(n)$ 값을 검색한다. 이웃한 3개의 노드 $h'(n)$ 값은 각각 2, 10, 3이므로 이번에도 현 시점에서의 최적인 {작업2:에이전트2}가 선택된다. 이러한 방식으로 탐색이 수행된 결과 작업 수행 예상시간이 가장 적을 것으로 예상되는 {작업1:에이전트1}, {작

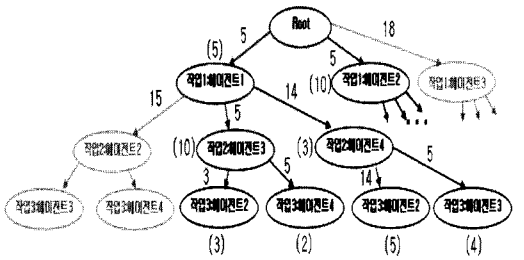
업2:에이전트2}, {작업3:에이전트3}이 선택된다.



[그림 10] 그리디 방식 적용 사례

(2) A* 알고리즘 방식

A* 알고리즘을 적용한 조정 에이전트를 통한 작업 할당의 경우 평가함수는 [수식 2]가 되며 할당되는 과정은 [그림 11]과 같다. 선택 단계에서는 작업 수행 예상시간과 실제 작업 수행시간을 고려한 평가가 이루어지며 탐색이 수행된다. 우선 조정 에이전트는 이미 정의된 정제조건에 대한 정제를 수행하며 [그림 11]의 경우 정제조건을 $g'(n)$ 값이 15이상일 때 정제한 결과로, 탐색 영역이 감소함을 알수 있다. 평가 함수에 의한 탐색은 [수식 2]를 적용한 실제작업수행시간과 작업수행예상시간의 합에 따라 선택된다. Root에서 정제된 조합을 제외한 {작업1:에이전트1}, {작업1:에이전트1}에서 평가함수를 적용한 값은 각각 10과 15가 되며 최소 비용이 예상되는 {작업1:에이전트1}이 선택된다. 이러한 반복적인 선택 과정으로 할당된 결과는 {작업1:에이전트1}, {작업2:에이전트3}, {작업3:에이전트2}가 된다. 이는 실제 에이전트가 작업을 수행하는 시간을 고려한 평가를 통한 할당이며 [수식 3]을 만족하는 A* 알고리즘을 적용한 결과이기 때문에 최적의 결과가 된다.



[그림 11] 제안하는 조정 에이전트 적용 사례

4. 실험 및 분석

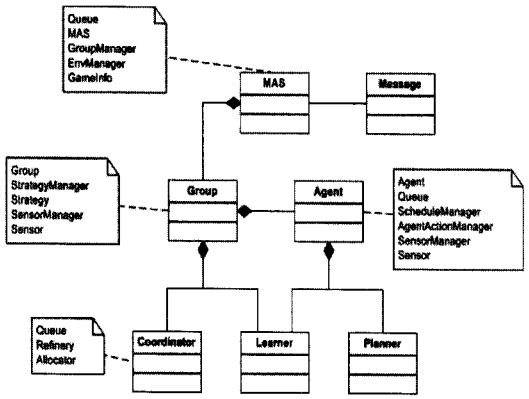
본 논문에서는 실시간 다중 에이전트 시스템 환경에서, 제안하는 조정 에이전트를 통한 작업 할당이 수행되었을 경우 최적화된 결과를 나타내는지 확인한다. 최적화 성능 평가방식은 동일한 환경에서 기존의 그리디 방식으로 할당이 수행된 경우와 A* 방식으로 할당이 수행된 경우의 최종 결과를 비교 분석한다.

4.1 실험 환경

(1) 시스템 구성

실험 환경은 다중 에이전트 시스템 구조로 설계된 MASE(Multi Agent System Engine)를 사용한다[그림 12]. MASE는 다중 에이전트 시스템의 특징인 학습, 계획, 조정 기능이 포함되어 있으며 실시간 환경에 적합한 엔진이다. MASE는 시스템, 그룹, 에이전트 3개의 계층적 구조로 설계되었다. 시스템 단계에서는 다수의 에이전트로 구성된 그룹을 관리하는 층으로 그룹 또한 여러 개가 존재할 수 있다. 시스템에서는 메시지 처리를 담당하며 외부로부터 전달된 메시지를 그룹에 전달하는 기능을 가지고 있다. 그룹 단계에서는 다수의 에이전트를 관리하고 전역적인 목표를 수행하기 위한 복합작업들을 학습과 조정을 통해 결정한다. 에이전트는 계획, 학습의 기본 기능을 가지고 있으며 현재 상태에서의 행동을 계획하고 실행한다. 수행한 결과는 학습에 반영되어 평가되며 이를 통한 학습이 가능하다. [그림12]는 모의실험 환경을 위한 시스템으로 Message는 통신 클래스간 정보를 교환하기 위한 기본 클래스이며 다중 에이전트 시스템에서 통신되는 모든 메시지는 이 클래스를 상속받는다. Group은 엔진에서 사용될 그룹을 정의하기 위한 최상위 클래스로 각 그룹마다 다른 기능과 속성을 정의하기 위해서는 Group을 상속 받아 정의한다. Agent는 에이전트의 기능을 처리하기 위한 기본 클래스로 다양한 에이전트를 구현하기 위해서는 이 클래스를 상속 받아 추가적인 기능을 정의한다. Learner는 에이전트와 그룹의 행동 결과에 따른 보상값을 통해 학습을 수행하며, 학습된 데이터를 이용하여 목표를 수행하기에 적합한 최적의 복합작업을 선택하는 기능이 있다. Learner에 의해 선택된 복합작업은 Coordinator를 통해 구성작업으로 나누어지며, 최적화 방식

에 따라 가장 적합한 에이전트에게 구성작업을 할당한다. 구성작업을 할당 받은 에이전트는 Planner를 통해 자율성을 기반으로 최상의 행동을 수행한다.



[그림 12] MASE 시스템 구조

실험에 사용하는 작업은 사전 정의되어 있다. 전역적인 목표를 수행하기 위한 복합작업은 에이전트가 수행하기 위한 다수의 구성작업들로 이루어졌으며, 에이전트가 구성작업들을 수행함으로써 복합작업이 완성된다. 에이전트와 그룹은 MASE에서 정의되며 생성한다. 에이전트는 그룹 관리자에 의해서 관리되며 고유의 기능과 역할은 사전 정의한다.

실험 시스템 환경은 Intel Pentium 4 (2.4GHz) CPU, 512MB RAM, Windows XP Professional SP2 운영체제 환경에서 개발 툴은 Borland JBuilder 2005를 사용하였으며, 모의실험은 MIDP Emulator 환경에서 실시하였다.

(2) 기본 설정

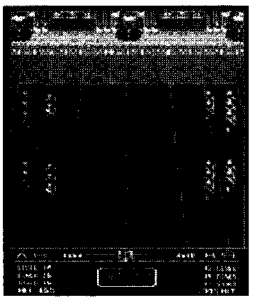
평가함수에 적용하기 위한 최적화 조건은 작업과의 거리가 가장 가까운 에이전트가 할당받는 것을 최적화된 결과로 설정한다. 거리에 해당하는 F(n) 값의 계산은 작업과 에이전트의 x, y좌표를 이용한 거리지수 계산방식을 적용한다.

조정 에이전트의 기본적인 정제조건은 작업과의 거리가 먼 에이전트가 작업을 할당받는 것은 비효율적이라는 가정하에 거리가 일정거리 이상의 에이전트는 정제되도록 한다. 그러나 기본적인 정제 조건만을 적용했을 경우에는 최적화 조건을 만족하더라도 에이전트의 고유 역할에 따른 정제가 필요한 상황이 발생할 수 있다. 실험에서는 사용자 정제조건을 적용하여 이 조건을 만족하는 에이전트는 최적화 조건과 상관없이 정제가 수행된다.

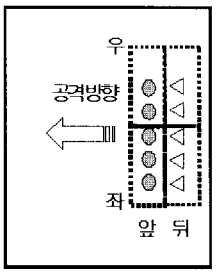
휴리스틱에 대한 추정 값은 실험 시나리오에 따라 A* 알고리즘을 만족하도록 적절하게 설정되어야 한다. 본 시나리오에서는 A* 알고리즘의 최적화 조건을 만족하는 (거리*0.7) 값을 휴리스틱 값으로 한다.

4.2 실험 내용

모의실험은 MASE를 기반으로 구현된 전략 시뮬레이션 게임에서 조정 에이전트를 실험 환경에 맞게 수정 및 보완하여 수행한다. 모의실험 시나리오는 [그림 13]과 같이 두 개의 그룹이 공격진형에 따라 전투를 실시하는 방식으로, 전투 중 모든 에이전트가 소멸하는 팀이 패하는 것을 기본으로 한다. 제한시간 안에 승패가 결정되지 않을 경우에는 에이전트가 많이 남아있는 그룹이 이기는 것으로 한다.



[그림 13] 모의실험 화면



[그림 14] 기본공격진형

각 그룹은 총 10개의 에이전트로 이루어졌으며, 에이전트의 종류는 근거리 공격을 담당하는 보병과 원거리 공격만 가능한 궁수가 존재한다. 그룹 전체의 공격진형을 나타내는 복합작업은 총 3개로 구성되며, 복합작업에 따른 에이전트의 수행 과정인 구성작업은 최대 4개로 설정한다 〈표 2〉.

그룹 인덱스	그 룹 설 명	에이전트의 최대개수	복합작업의 최대개수	구성작업의 최대개수
0	보병 5, 궁수 5로 이루어진 그룹 0	10	3	4
1	보병 5, 궁수 5로 이루어진 그룹 1	10	3	4

〈표 2〉 모의실험 구성

공격진형은 그룹 차원에서 수행되는 복합작업을 의미하며 기본 공격진형은 [그림 14]와 같이 근거리 공격의 보병 에이전트가 앞에 위치하고 원거리 공격의 궁수 에이전트가 뒤에 위치한 상황에서 중앙으로 전진하면서 공격을 취하는

진형이다. 공격진형에는 기본 공격진형과 측면공격진형, 방사형 공격진형이 있다.

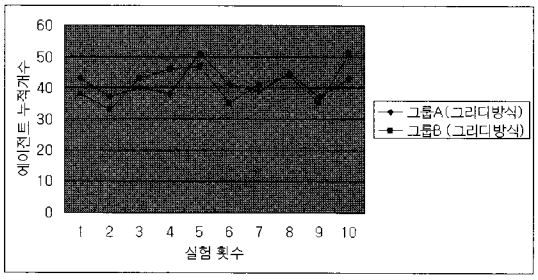
그룹 전체에게 할당되는 공격진형은 각 에이전트가 수행하는 구성작업으로 나누어진다. 구성작업은 에이전트의 위치에 따라 평가된다. 그리디 방식의 경우 진형위치좌표에 가까운 에이전트에게 높은 우선순위를 부여함으로써 할당이 수행되고, A* 방식은 진형을 이루기 위한 이동회수에 따른 최적화 평가함수를 적용함으로써 할당이 수행된다.

정제조건은 진형과 거리가 먼 에이전트가 정제되는 것을 기본 정제조건으로 한다. 사용자 정제조건은 이미 다른 전략을 수행하고 있는 경우, 보병 에이전트나 궁수 에이전트가 모두 소멸되었을 경우, 적과 교전중인 에이전트는 정제되는 것으로 한다.

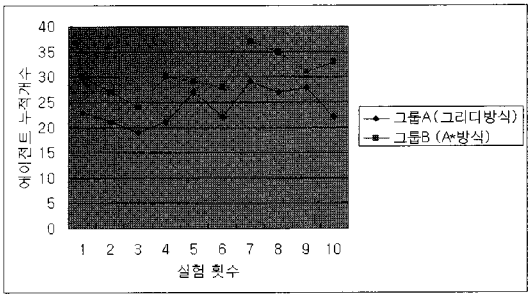
실험은 그리디 방식과 A* 방식의 성능 비교를 위해 두 그룹 모두 그리디 방식을 적용한 실험1과 한쪽 그룹만 A* 방식을 적용하는 실험2로 진행된다. 실험 결과는 10회의 전투를 실시하여 각 상황 종료 후 남아있는 에이전트의 누적개수를 비교하는 방식으로 한다. 에이전트의 누적개수가 많은 것은 승률이 높은 것이며, 모의실험에서 궁극적인 목표를 수행하기 위한 작업 할당이 효과적으로 수행되었음을 의미한다. 또한 성능 평가의 신뢰도를 위해 같은 방식으로 10회 실험을 실시한다.

실험1은 두 그룹 모두 그리디 방식으로 작업을 할당하며 [그림 15]에서와 같이 에이전트의 누적개수 차이가 일정하지 않다. 이는 승률이 일정하지 않았거나 비슷한 성능을 가지고 있기 때문이다.

그러나 실험2는 [그림 16]과 같이 그룹B에게 A* 방식을 적용함으로써 그룹A의 그리디 방식과 비교하여 에이전트의 누적개수가 평균 25% 향상되었음을 알 수 있다. 이는 그리디 방식과 비교하여 A* 방식으로 할당이 수행되었을 경우 높은 승률을 나타내는 것을 의미한다.



[그림 15] 실험 1 수행결과



[그림 16] 실험 2 수행결과

5. 결론 및 향후 연구계획

본 논문에서 제안하는 작업 할당 방안은 중앙 집중적 작업 할당 방안인 기존의 그리디 방식에서 발생하는 최적화 보장 문제를 A* 알고리즘을 적용하여 보완하였다. 작업 할당을 최소 비용 탐색 방법인 A* 알고리즘을 적용하여 수행함으로써 최적화 조건을 만족하는 에이전트에게 작업이 할당된다. 또한 A* 알고리즘으로 작업 할당이 수행될 경우 발생하는 할당 지연에 대한 문제점은 정제 기능으로 방지하였다. 정제는 정제 조건을 만족하는 작업과 에이전트를 사전에 제거함으로써 탐색공간을 최소화한다. 실시간 동적인 환경에 대한 작업의 재할당 요구는 유동적인 할당 방식으로 보완하였다. 조정 에이전트는 재할당 요구에 따른 처리를 위해 할당 지연이 발생하는 상황에서 그리디 방식으로 할당 방식을 변경함으로써 빠른 할당이 가능하다. 모의실험은 그리디 방식과 비교하여 제안하는 A* 알고리즘 방식이 최적화된 작업 할당 결과로 인해 평균 25%정도 성능이 향상되었음을 입증하였다. 제안하는 조정 에이전트는 실시간 다중 에이전트 시스템 환경에 적합한 작업 할당 방안으로 최적의 작업 할당이 수행됨에 따라 작업의 효율성 증가와 이에 따른 시스템 성능 향상을 기대할 수 있다. 조정 에이전트는 실시간으로 급변하는 게임과 같은 환경에서 작업 할당 수행 역할을 담당하는 엔진에 적용할 수 있다.

향후 연구 계획으로는 A* 알고리즘 적용시 발생하는 탐색 공간 증가를 방지하기 위한 연구로 정제 기능을 자동화하는 방법을 계획 중이다. 정제의 자동화는 학습을 통해 자율적으로 수행되는 정제 기능을 의미하며, 이는 탐색 속도를 향상시킬 것으로 예상된다. 또한 개량된 A* 알고리즘들을

적용하였을 때는 알고리즘의 성능이 어떻게 향상되는 지에 대한 비교분석실험도 진행되어야 한다.

참고문헌

[1] Katia P. Sycara, "Multiagent Systems", AI Magazines(98) Summer, pp.79-92, 1998.

[2] Steve Rabin, "Squad Tactics: Team AI and EmergentManeuvers", AI Game Programming Wisdom 1, Charles River Media, pp.233-246, 2003.

[3] Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. "Network Flows: Theory, Algorithms, and Applications", Prentice Hall, Upper Saddle River, New Jersey, pp.309-312, 1993.

[4] Gerkey, B. P, Mataric, M. J. "A Framework for Studying Multirobot Task Allocation", Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II, Proceedings of the 2nd International Naval Research Laboratory Workshop on Multi-Robot Systems (Washington, DC, March 17 - 19), 2003.

[5] Durfee, E. H. "Scaling up agent coordination strategies", Computer, Volume 34, Issue 7, pp.39-46, 2001.

[6] Malone, T. W. "The Interdisciplinary Study ?of Coordination", ACM computing surveys, Volume 26,?Issue 1, pp.87-119, 1994.

[7] Steve Rabin, "Goal?Directed Behavior Using Composite Task", AI Game Programming Wisdom 2, Charles River Media, pp.237-245,?2003.

[8] Gerkey, B. P, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems", The International Journal of Robotics Research, Vol. 23, No. 9, pp.939-954, 2004.

[9] Kristina Lerman, Chris Jones, Aram Galstyan, Maja J Matar?c, "Analysis of Dynamic Task Allocation in Multi-Robot Systems", International Journal of Robotics Research Volume 25, Issue 3, pp.225-241, 2006.

[10] Steve Rabin, "Basic A* Pathfinding Made Simple", AI Game Programming Wisdom 1, Charles River Media, pp.105-113,?2003.

[11] Stuart J. Russell and Peter noving, "Artificial Intelligence : A Modern Approach", Prentice-Hall, pp.92-121, 1995.

박 재 현 (Jaehyun Park)

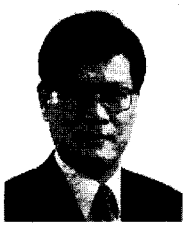
1999.2 동국대학교 컴퓨터공학과 졸업 (공학사)
2006.8 동국대학교 일반대학원 컴퓨터공학과 졸업(공학석사)
2007.3 ~ 현재 (주)Liho Communication

관심분야 : 게임 인공지능, 게임 프로그래밍, 데이타베이스

조 경 은 (Kyungeun Cho)

1993.2 동국대학교,전자계산학과(공학사)
1995.2 동국대학교, 컴퓨터공학과 대학원(공학석사)
2001.8 동국대학교, 컴퓨터공학과 대학원(공학박사)
2002.3 ~ 2003.2 안양대학교 디지털미디어학부 강의전담교수
2003.3 ~ 2003.8 영산대학교 게임공학과 전임강사
2003.9 ~ 2005.8 동국대학교 정보산업대학 컴퓨터멀티미디어공학과 전임강사
2005.9 ~ 현재 동국대학교 영상미디어대학 게임멀티미디어공학과 조교수

관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 멀티미디어 정보처리



엄 기 현 (Kyhyun Um)

1975년 서울대학교 공과대학 응용수학과 공학사
1997년 한국과학기술원 전산학과 이학석사
1994년 서울대학교 대학원 컴퓨터공학과 공학박사
1978년 3월~2005년 8월 동국대학교 컴퓨터멀티미디어공학과 정교수
2005.9 ~ 현재 동국대학교 영상미디어대학 게임멀티미디어공학과 교수
1995.3~1999.2 동국대학교 정보관리처장 역임
1998.9~2000.8 데이터베이스 연구회 운영위원장
1998.8~2000.7 한국정보과학회 데이타베이스연구회운영위원장,
1998.12~2001.12 한국 멀티미디어학회 부회장
1999.4~현재 Int. Conf. on Database Systems for Advanced Applications Steering Committee 위원
2001.3~2003.2 동국대학교 정보산업대학 학장 역임
2001.1~2002.12 한국정보과학회 논문지편집부위원장(데이타베이스담당)
2004.1~2005.2 한국 게임학회 부회장
2005.3~현재 한국 게임학회 자문위원
2004.1~2005.12 한국 멀티미디어학회 자문위원
2006.1~2006.12 한국멀티미디어학회 수석부회장
2007.1~현재 한국멀티미디어학회 회장

관심분야 : 게임시스템 설계, 멀티미디어 데이터베이스, 멀티미디어 정보시스템