석 사 학 위 논 문

# Cellular Cosmos : 셀 수준의 인공개체 진화 프레임워크 구조 및 응용

Cellular Cosmos : Architecture and Application of Cell-level Evolution Framework for Artificial Individuals



2015년

한성대학교 대학원

정보통신공학과

정보통신공학전공

정 보 선

석 사 학 위 논 문
지도교수 정성훈

# Cellular Cosmos : 셀 수준의 인공개체 진화 프레임워크 구조 및 응용

Cellular Cosmos : Architecture and Application of Cell-level Evolution Framework for Artificial Individuals



2015년 12월 일

한성대학교 대학원
정보통신공학과
정보통신공학전공
정 보 선

석 사 학 위 논 문
지도교수 정성훈

# Cellular Cosmos : 셀 수준의 인공개체 진화 프레임워크 구조 및 응용

Cellular Cosmos : Architecture and Application of Cell-level Evolution Framework for Artificial Individuals

위 논문을 공학 석사학위 논문으로 제출함

2015년 12월 일


한성대학교 대학원
정보통신공학과
정보통신공학전공
정 보 선

정보선의 공학 석사학위논문을 인준함

2015년 12월 31일

심사위원장 노 광 현 인

심 사 위 원 정 성 훈 인

심 사 위 원 신 성 인

# 국 문 초 록

## Cellular Cosmos : 셀 수준의 인공개체 진화 프레임워크 구조 및 응용

**한성대학교 대학원**
**정보통신공학과**
**정보통신공학전공**
**정 보 선**

지구상에 존재하는 모든 생물들은 자신이 서식하는 환경에서 살아남기 위하여 변화해 왔으며 지금 현재의 모습을 갖게 되었다고 한다. 우리는 컴퓨터상에서 이러한 메커니즘을 분석 및 이해하고 이를 통하여 밝혀진 메커니즘을 생물학적 및 공학적 응용을 할 수 있도록 하기 위하여 컴퓨터상에 모사할 수 있는 진화프레임워크를 제안하고 간단한 인공개체-먹이 모델을 적용하여 시뮬레이션 해보았다. 또한 인공개체의 행동진화에 있어서 자신의 행동을 기억하는 개체와 그렇지 않은 개체의 차이를 실험을 통해 알아보았다.

셀 우주는 2차원의 격자 셀 공간으로 구성되어 있고 이 셀 상에 인공개체와 먹이가 존재할 수 있다. 인공개체는 자신의 행동을 결정하는 행위결정 로직프레임을 갖고 있으며 이를 통해 이웃 셀의 먹이 상황에 대하여 자신의 행동을 결정하여 행동할 수 있다.

시뮬레이션 초기의 인공개체들은 임의의 내부로직을 갖고 태어나 먹이를 잘 먹지 못한다. 하지만 동일한 세대에서 비교적 먹이를 잘 먹은 인공개체가 잘 살아남아 번식하여 자신의 변이된 내부로직을 갖는 자녀를 생성하게 된다. 그렇기 때문에 세대가 흐를수록 먹이를 더욱 잘 먹는 인공개체가 많이 살아남게 되고 진화가 이루어진다. 우리는 기억회로가 있는 인공개체와 없는 인공개체 중 어느 인공개체가 먹이를 잘 먹도록 진화를 하는지 실험을 통해 알아보았다.

기억회로가 없는 인공개체는 오로지 먹이에 대한 입력만 받기 때문에 먹이환경이 바뀌지 않는 상황에선 여러 가지 가능한 행동 중 단순히 하나만을 결정해서 반복 수행한다. 하지만 행위결정 로직프레임에 기억회로를 추가함으로서 인공개체는 동일한 환경에서도 다양한 행동을 수행할 수 있게 된다. 행위결정 로직프레임의 기억회로를 통해 인공개체가 과거에 한 자신의 행동을 되먹임하여 다시 입력으로 받게 된다. 그렇기 때문에 주변의 먹이 상황이 이전과 동일하더라도 이전에 결정했던 행동에서 유리한 행동을 선택할 수 있다. 이를 통하여 기억회로를 갖는 인공개체는 먹이를 더욱 잘 먹을 수 있게 된다.

본 연구에서 제시한 시뮬레이션 프레임워크를 통해 아직까지 풀리지 않은 생물의 진화과정의 궁금증을 풀거나 진화적으로 접근하여 풀고자 하는 공학적 문제를 시뮬레이션을 통해 풀고자 하는 곳에 응용할 수 있다.



【주요어】 인공개체, 먹이, 진화, 프레임워크, 변이, 기억회로, 단성생식

# 목 차

# 표 목 차

# 그 림 목 차

# I. 서 론

## 1.1 연구 배경

지구상에 존재하는 모든 생물들은 과거로부터 현재까지 자신이 서식하는 환경에서 살아남기 위하여 적응해 왔으며 지금의 모습을 갖게 되었다고 한다[Davis, P., 2000 : Staguhn, G., 2004]. 이렇게 생물들이 환경에 적합한 형태로 변화해 가는 것을 진화[Mayr, E., 2001]라고 하는데 우리는 이러한 진화 메커니즘을 컴퓨터상에서 분석하며 이해하기 위하여 시뮬레이션 가능한 진화 프레임워크를 제안한다. 진화 메커니즘의 가장 기본은 다윈이 주장한 자연선택[Darwin, C., 1859]이 있다. 자연선택이란 어떠한 환경에 놓인 생물체의 종들이 그 환경에 적합한 형태를 가지고 있다면 그렇지 못한 종 보다 살아남을 확률이 높다는 것이다. 허버트 스펜서에 의해 처음 사용된 언어인 적자생존은 자연선택의 메커니즘이며 환경에 적합한 개체가 잘 살아남는다는 원리를 설명하고 있다. 환경에서 잘 살아남은 개체가 더 많이 번식하고 더 많은 변종을 만들어 이전 세대보다 더욱 유리한 형태로 변화하기 때문이다[Losos, J. 등, 2013].

현재 진화 메커니즘을 사용한 다양한 연구가 여러 분야에서 연구되고 있다[Torresen, J., 2004 : Minsky, M., 1961]. 대부분의 진화 알고리즘의 연구의 목적은 공학적 문제 해결에 있어서 해답을 최적화 하는데 중점을 두고 있다[Mitchell, M., 1998 : Mitchell, M., 1995]. 그리고 다양한 최적화 문제에 대하여 좋은 성능을 내고 있다[Backer, B. B. 등, 2003 : Michalewicz, Z. 등, 1992]. 이러한 문제 해결에 대한 진화 알고리즘의 성능을 통해 우리는 진화 알고리즘에 있어서 진화적 요소가 적용 가능함을 알 수 있으며 앞으로도 더욱 다양한 분야에서 사용 가능할 것이다[Müller, V. C. 등, 2014].

진화하기 위해 필요한 요소 외에도 개체가 환경에 잘 적응하기 위하여 필요한 다른 요소는 바로 기억이다[Sherry, D. 등, 1987]. 세대간의 기억의 전달, 즉 부모로부터 물려받은 기억과 지식이 있었기 때문에 현재의 진보된 사회와 문명이 존재할 수 있었고 또한 앞으로의 사회가 발전 해 나아갈 수 있게 된다[Jedlowski,

P., 2001]. 만약 세대간의 기억의 전달이 없었다면 새로운 세대가 시작될 때 마다 인류는 과거에 했던 실수를 반복하였을 것이다. 또한 새로운 발견을 하더라도 다음 세대에게 그것에 대한 역사를 남겨주지 않기 때문에 발견이 무의미 하게 될 것이다. 하지만 인류는 과거의 역사와 새로운 발견을 기록이나 다양한 방법을 통해 다음 세대에게 넘겨주기 때문에 현재의 진보된 사회가 만들어 질 수 있었다.

많은 학자들의 연구에도 불구하고 아직까지 진화란 무엇인지, 또한 어떻게 이루어 졌는지 명확히 밝혀진 것은 없다[Jackson, W., 1996 : Denton, T. 등., 1986]. 그렇기 때문에 우리는 진화 대상의 진화 과정을 시뮬레이션을 통해 관찰 가능하고 분석 가능한 프레임워크를 제안하였다. 이 프레임워크를 통해 셀 상에 살아가는 인공개체가 진화 메커니즘과 기억을 통해 자신이 살아가는 환경에 적합한 형태로 진화할 때 인공개체의 행동이 진화 전과 후에 어떻게 달라졌는지 시뮬레이션을 통해 분석해 보았다.

## 1.2 연구 내용

우리는 1.1 절에서 설명한 진화 메커니즘을 컴퓨터 상에서 구현 가능한 시뮬레이션 프레임워크인 셀 우주를 제안하였다. 셀 우주는 2차원 셀 격자 공간 내에 살아가는 인공개체가 먹이를 먹고 단성생식과 변이를 통하여 자녀를 낳고 자녀들이 환경에 적합한 형태로 진화하는 시뮬레이션 프레임워크이다. 인공개체들은 이웃 셀의 먹이상황을 인식하여 입력을 받는 행위결정 로직프레임을 갖고 있다. 또한 행위결정 로직프레임은 내부로직을 갖고 있으며 이 내부로직을 통해 먹이입력에 대한 출력을 내보내게 된다. 그리고 이 출력이 인공개체의 행동을 결정한다.

셀 우주에서 처음 태어난 인공개체들은 임의의 내부로직을 갖기 때문에 먹이를 잘 먹는 행동을 하지 못한다. 하지만 세대가 흐르며 인공개체들은 번식을 하고 자녀들에게 자신의 변이된 내부로직을 상속시켜 준다. 부모의 변이된 내부로직을 상속받은 자녀는 부모와 다른 행동을 하게 되는데 만약 이 행동이 부모보다 먹이를 잘 먹는 행동이라면 자녀 인공개체들이 더 많이 살아남게 된다. 이러한 과정이 반복적으로 일어나며 시간이 흐를수록 인공개체들은 먹이를 잘 먹도록 진화한다.

단순히 먹이에 대한 입력만 받는 행위결정 로직프레임은 오로지 먹이에 대해서만 반응을 하기 때문에 과거와 동일한 먹이환경에 놓인다면 단순히 동일한 행동밖에 할 수 없다는 한계가 있다. 우리는 이러한 한계점을 개선시키기 위하여 행위결정 로직프레임에 기억회로를 추가하였다. 기억회로는 인공개체가 과거에 한 행동을 되먹임 받아 현재의 입력으로 받는 구조를 가지고 있다. 그렇기 때문에 인공개체들은 동일한 먹이 환경에 놓여 있더라도 과거에 자신이 한 행동을 기억하여 다른 행동을 할 수 있게 된다.

우리는 기억회로가 없는 인공개체와 기억회로가 있는 인공개체를 시뮬레이션 하였다. 먼저 기억회로가 없는 행위결정 로직프레임을 검증하기 위하여 몬테카를로 방법[Mackay, D. J., 1998]을 통해 우리가 제안한 행위결정 로직프레임의 동작성을 검증하였다. 이를 위하여 우리는 임의로 내부로직들을 여러번 반복 생성하였다. 그리고 이 내부로직들이 인공개체가 가질 수 있는 모든 먹이환경에 대하여 먹이를 먹는 방향으로 행동할 수 있는 출력을 내는지 확인하였다. 또한 셀 우주에서 제안하는 진화 프레임워크를 검증하기 위하여 인공개체의 적합도 평가를 통한 진화를 시뮬레이션 하여 보았다. 적합도 평가는 인공개체가 가질 수 있는 모든 먹이환경에 대해 분석한 후 그 내용을 기준으로 시뮬레이션 상의 인공개체의 행동과 비교하여 보았다. 그리고 시뮬레이션 상의 인공개체가 분석된 행동대로 진화하는지 확인하였다.

기억회로가 없는 인공개체와 있는 인공개체의 차이를 확인하기 위하여 다양한 먹이구조를 만든 후 두 인공개체를 따로 진화시켜보았다. 먼저 단순한 먹이구조에서 진화켜 보았다. 단순한 먹이구조에서는 오히려 기억회로를 갖는 인공개체가 불리하게 작용할 수 있는데 이는 행위결정 로직프레임의 구조가 더욱 복잡해져서 탐색공간의 크기가 커지기 때문에 진화에 영향을 미칠 수 있기 때문이다. 그 다음으로 복잡한 먹이구조에서 진화시켜 보았다. 이는 기억회로가 없는 인공개체의 한계를 알아보기 위함이었다. 인공개체는 동일한 먹이환경에 대해 동일한 행동을 할 수 밖에 없으므로 복잡한 먹이구조를 선택하여 실험하였다.

또한 기억회로는 인공개체가 몇 단계의 과거 행동을 기억하는지에 따라 크기가 달라진다. 이론상으로는 많은 단계의 과거 행동을 기억할수록 더욱 복잡한 행동이 가능하기 때문에 좋아야 한다. 하지만 위에서 언급한 탐색공간의 문제가 발생

하게 된다. 그렇기 때문에 우리는 특정한 먹이구조를 선택하여 서로 다른 단계만큼의 행동을 기억하는 두 인공개체들을 동시에 진화시켜 어떠한 인공개체가 더욱 잘 진화하는지 확인하였다. 또한 모든 인공개체들을 동시에 진화시켜 가장 좋은 인공개체가 무엇인지 확인한 후 이 전 실험과 결과를 비교하였다.

우리는 위에서 설명한 실험을 통하여 셀 우주의 프레임워크를 검증하였으며 인공개체가 갖는 행위결정 로직프레임의 동작성을 확인하였다. 그리고 이 실험의 결과를 통해 기억회로가 없는 인공개체와 있는 인공개체를 각각 진화시켜 보았다. 두 인공개체들의 행동이 서로 어떻게 다른지 다양한 먹이구조의 실험을 통해 분석하였다. 그리고 기억이 있는 인공개체들이 자신의 행동을 얼만큼 기억하는 것이 가장 유리한지 확인하였다.

# II. 관 련 연 구

본 연구에서 제시한 셀 수준의 인공개체 시뮬레이션 프레임워크와 동일한 연구는 없다. 다만, 목적이 유사하거나 시뮬레이션 환경이 유사한 기존의 연구들이 있다. 본 장에서는 우리 연구와 유사한 기존의 연구를 소개한다. 먼저 세포자동자이다[Langton, C. G., 1984 : Sakar, P., 2000]. 세포 자동자는 셀공간에서 동일한 몇 가지 규칙으로 다음 상태를 결정하며 스스로 동작한다. 우리 연구에서는 이러한 셀 공간과 유사한 시뮬레이션 공간을 사용한다. 다만 세포자동자에서는 각각의 셀이 하나의 상태이나 우리는 각각에 셀에 먹이나 인공개체가 존재할 수 있다. 다음으로 진화과정을 모사해서 최적화에 이용하는 유전자 알고리즘이 있다[Mitchell. M., 1988]. 이는 진화의 특성을 모델링해서 공학적 최적화알고리즘을 만든 것이다. 우리 연구와 직접적인 관련은 없으나 인공개체가 자식을 낳아서 진화하는 특성측면에서 유사한 측면이 있다.

## 2.1 세포 자동자

세포 자동자(Cellular Automata)는 단성생식을 하는 인공 시스템을 목표로서 인공 생명체[Langton, C. G., 1989 : Langton, C. G., 1997]의 아버지라 불리는 존 폰 노이만에 의해 처음 만들어 졌다[Sarkar, P., 2000]. 세포 자동자는 무한한 1차원 혹은 2차원의 격자구조를 갖는 셀들이 자신의 이웃 셀들의 상태에 대하여 상태변화를 한다. 이웃 셀들이란 자기를 중심으로 주변에 존재하는 셀들을 의미하며 경우에 따라 얼마만큼의 셀을 이웃으로 설정할지 정할 수 있다. 이웃을 어디까지 설정 하느냐에 따라 이웃의 종류가 다른데 이중 4개의 가장 가까운 셀을 이웃으로 정의한 '폰 노이만의 이웃'이 있고 8개의 가장 가까운 셀을 이웃으로 정의한 '무어의 이웃'이 대표적인 예이다. 그림 1(a)는 폰 노이만의 이웃을 보여주고 있고, 그림 1(b)는 회전한 폰 노이만의 이웃을 보여주고 있으며 그림 1(c)는 무어의 이웃을 보여주고 있다. 세포 자동자의 시간 흐름은 동기식 이산시간이다. 그렇기 때문에 모든 셀들의 상태

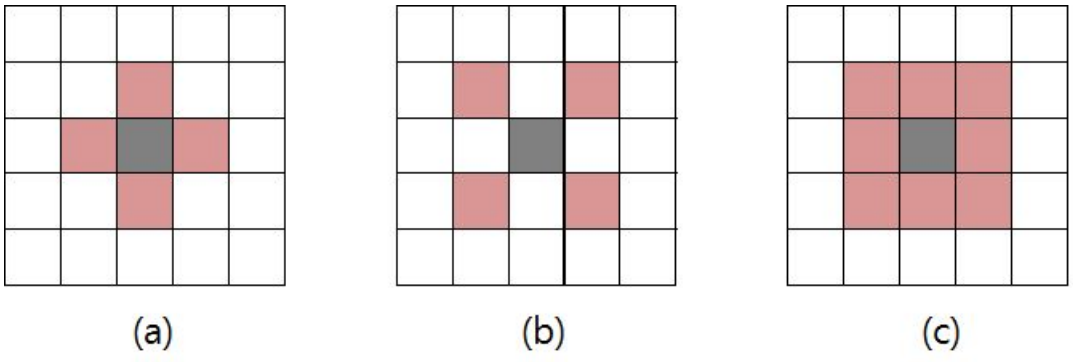



그림 1. 이웃의 종류 (a) Von Neumann neighborhood (b) Rotated Von Neumann neighborhood (c) Moore neighborHood

변화가 동일한 시간에 이루어지며 각 이산 시간 단계가 하나씩 증가 하는데 하나씩 증가하는 시간을 세대라고 한다[Kari, J., 2005]. 세포 자동자의 셀들은 '살아있음' 혹은 '죽어있음' 과 같은 유한한 상태를 가질 수 있다. 또한 모든 셀들은 동일한 규칙을 따르며 상태를 변화한다. 세포 자동자를 기반으로 다양한 연구가 이뤄졌는데 그 중 세포 자동자를 세상에 각광받게 한 연구는 1970년 존 콘웨이가 제안한 라이프 게임(Game of Life)이다[Jarkko, K., 2013].

라이프 게임은 2차원의 무한한 셀 격자 공간 내에 셀들이 살아있음(검정색) 혹은 죽어있음(흰색) 이라는 두 개의 상태만을 가지고 있으며 셀 이웃은 무어의 이웃을 기준으로 하고 상태 변화 규칙은 매우 단순한 4개의 규칙을 따른다. 규칙은 세포 자동자와 마찬가지로 모든 셀들에 동일하게 적용된다. 시간 흐름 또한 세포 자동자와 마찬가지로 동기식 이산시간이기 때문에 모든 셀의 상태 변화가 동일한 세대에 이루어 진다.

규칙 1. 살아있는 셀 중 1개 이하의 이웃 셀이 살아있을 경우 이 셀은 고립되어 죽는다.

규칙 2. 살아있는 셀 중 2개 혹은 3개의 이웃 셀이 살아있을 경우 이 셀은 살아남는다.

규칙 3. 살아있는 셀 중 4개 이상의 이웃 셀이 살아있을 경우 이 셀은 포화되어 죽는다.

규칙 4. 죽어있는 셀 중 3개의 이웃 셀이 살아있을 경우 이 셀은 살아난다.

라이프 게임의 간단한 규칙들을 통해 셀들이 매우 창발적이며 다양한 패턴을 만들어 내기 때문에 처음 라이프 게임이 대중에게 소개 되었을 때 많은 호응을 얻었다[Kari, J., 2005]. 다양한 패턴들 중에는 동일한 패턴에 멈춰 계속하여 살아가는 패턴, 일정한 주기를 가지고 반복하는 패턴, 그리고 방향을 갖고 이동하는 패턴 등이 있다. 그림 2(a)는 한 패턴에 멈춰 계속하여 살아가는 패턴(still life) 중 하나이다. 4개의 셀이 뭉쳐서 정사각형의 구조를 가지고 있고 각 셀 하나하나가 가지는 살아있는 이웃 셀의 수는 3개이고 이것은 라이프 게임의 규칙 2번을 만족시키기 때문에 살아있는 셀들은 다음 세대까지

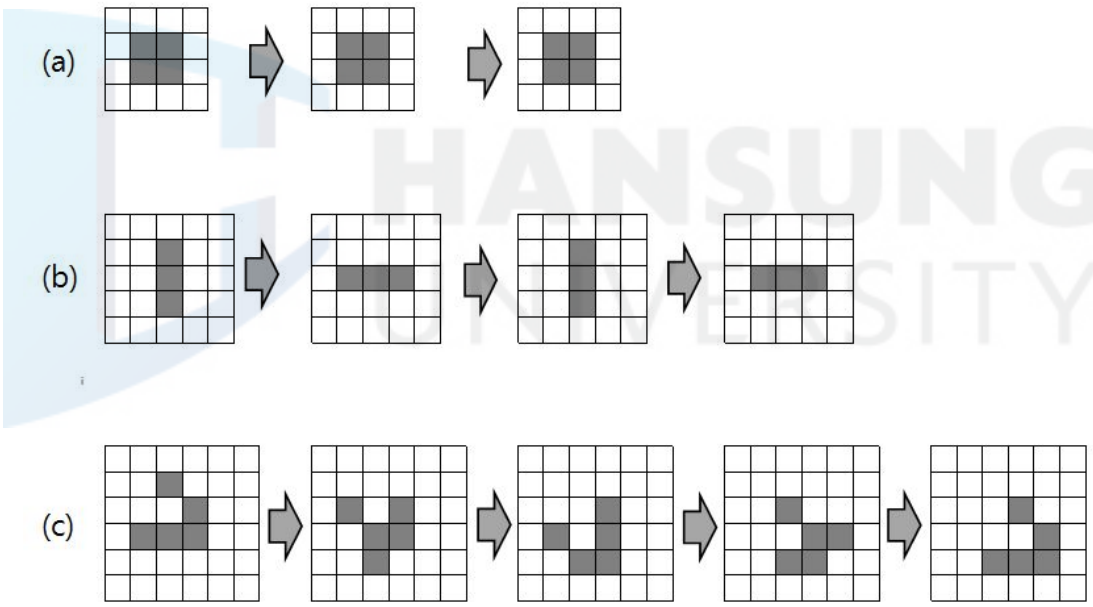



그림 2. 라이프 게임의 다양한 패턴 (a) still life (b) oscillator (c) glider

살아남게 된다. 그리고 외곽의 다른 죽어있는 셀들은 4번 규칙을 만족하지 못하기 때문에 살아나지 못한다. 그렇기 때문에 다음 세대에도 똑같은 상태를 갖고 있으며 외곽의 다른 셀이 어떠한 조건에 의해 살아나지 않는 이상 이 패턴은 무한히 유지된다. 그림 2(b)는 동일한 패턴을 반복하는 패턴(oscillator)중 하나이다. 살아있는 셀들 중 가운데 있는 셀은 2번 규칙을 만족시키기 때문에 다음 세대에도 살아남는다. 가운데 있는 셀을 중심으로 수직과

수평 방향에 살아있는 셀 2개가 있고 죽어있는 셀 2개가 있다. 살아있는 셀의 경우 이웃 셀들 중 살아있는 셀이 1개씩 밖에 없으므로 해당 셀은 1번 규칙에 의해 죽게 된다. 그리고 죽어있는 셀의 경우 이웃 셀들 중 살아있는 셀이 3개가 있으므로 4번 규칙에 의해 살아나게 된다. 이러한 조건이 매 세대 반복되기 때문에 그림 2(b)와 같은 패턴을 반복하게 된다. 그림 2(c)는 방향을 갖고 이동하는 패턴(glider)중 하나이다. 이 패턴은 4개의 패턴으로 주기를 이루고 있으며 still life와 oscillator와 마찬가지로 동일한 규칙에 의해 발생한다. 하지만 이 둘보다 더욱 흥미로운 것은 4개의 패턴을 반복하며 특정한 방향으로 이동하기 때문이다. 그림 2(c)의 경우 오른쪽 아래 방향으로 이동하는 glider의 경우를 보여주고 있다. 이와 같이 다양한 패턴 그리고 방향을 가지고 이동하는 패턴 등 여러 가지 셀의 상태에 의해 still life가 glider와 충돌할 경우가 생길 수 있으며 그 충돌로 인해 또 다른 창발적인 패턴이 발생할 수도 있다. 이처럼 여러 패턴을 적용하여 흥미로운 패턴들 만들어 낼 수 있다. 위에서 설명한 세 종류의 패턴 외에 더욱 흥미로운 패턴과 여러 패턴들을 동시에 적용하여 매우 신비로운 패턴을 만들어 나가는 패턴 또한 많이 있다 [Jarkko. K., 2013].

## 2.2 유전자 알고리즘

생물이 진화하기 위해 가장 중요한 요소는 생식, 변이, 교차 및 선택[Wright, S., 1932] 이다. 이러한 메커니즘을 다른 종목에 접목시켜 쉽게 시뮬레이션 가능하도록 컴퓨터상에 구현하는 연구는 많이 시도되었는데 그 중 가장 널리 알려져 있는 연구는 유전자 알고리즘이다[Mitchell, M., 1998 : Whitley, D., 1994]. 유전자 알고리즘은 어떠한 문제에 대한 해답을 최적화 시키는데 많이 사용된다 [Mitchell, M., 1995]. 이와 같은 문제를 최적화 문제라 하며 최적화 문제란 수학이나 과학 등에서 모든 경우의 수를 찾을 수 없는 문제에 적용하여 가장 최적의 해답을 찾는 방법에 대한 문제이다. TSP(Traveling Salesman Problem) [Dorigo, M. 등, 1997 : Noraini, M. R. 등, 2011]는 다양한 최적화 문제 중 하나이다. TSP란 어떠한 세일즈맨이 물건을 팔기 위해 여러 도시를 한번 씩 방문하여 모든

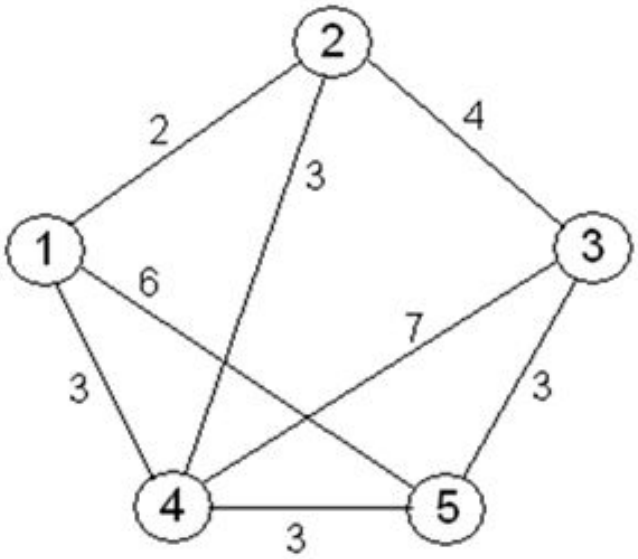



그림 3. TSP의 예제

도시를 거쳐 다시 원래 출발했던 도시로 돌아오는 경우의 수를 다루는 문제이다 [Flood, M. M., 1956]. 이 문제의 요소인 각 도시 간의 이동 시간과 도시 간 이동할 수 있는 경로들을 고려하여 어떠한 경로를 선택하여 이동하는 것이 가장 최적인지를 찾아야 한다. 그림 3에서는 TSP의 하나의 예를 보여주고 있다. 각 노드는 TSP의 도시 위치를 보여주고 있고 각 선분은 이동할 수 있는 경로를 보여주고 있고 경로와 노드 사이의 숫자는 해당 경로를 통해 갈 경우 걸리는 시간을 보여주고 있다.

위에서 설명한 TSP와 같은 최적화 문제를 위해 유전자 알고리즘을 통해 풀 수 있다. 유전자 알고리즘은 풀고자 하는 문제에 대한 해답의 경우의 수를 이진값의 유전자를 갖는 염색체로 표현한다[Mitchell, M., 1995]. 유전자 알고리즘의 순서도는 그림 4와 같다. 유전자 알고리즘이 시작되면 임의의 유전정보를 갖는 N개의 염색체들을 생성한다. 이 염색체들의 유전자의 적합도를 평가를 통해 부모 선택이 이루어지며 선택된 부모는 생식을 통해 자녀를 생성한다. 이 과정까지가 한 세대이며 각 세대가 끝날 때 마다 프로그램의 종료 조건을 확인하게 된다. 만약 조건을 만족한다면 프로그램은 종료될 것이고 그렇지 못하다면 세대가 하나 증가하며 다시 염색체들의 적합도를 평가하게 된다.

유전자 알고리즘에서 염색체들은 이진 비트열를 가지고 있는데 이 유전자에 의해 해당 염색체가 문제의 해답의 어떠한 경우를 보여주고 있는지 파악할 수 있

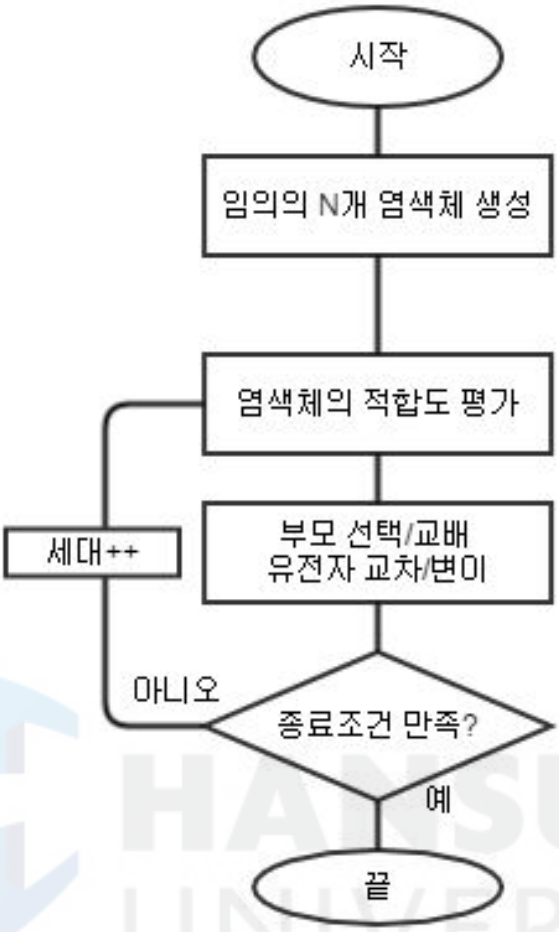



그림 4. 유전자 알고리즘의 순서도

다. 유전자 알고리즘을 처음 시작하면 사용자가 설정한 수의 염색체를 생성 한다. 이 염색체들은 임의의 유전자를 갖기 때문에 대부분은 문제의 해답과 근접하지 못한 유전자를 가지고 있다. 유전자들이 얼마나 좋은지 혹은 좋지 못한지 판단하기 위해 적합도 함수를 통해 적합도 평가를 한다. 적합도 평가를 통해 각 염색체들은 점수를 얻는데 점수가 높을수록 문제에 대해 좋은 유전자를 가지고 있음을 알 수 있다. 유전자 알고리즘은 각 세대가 지날 때 마다 두 부모를 선택하여 두 자녀를 생성하는데 이 때 적합도 점수가 높을수록 부모로 선택될 가능성이 높다. 부모 선택 기법은 다양한데 그 중 가장 널리 사용되는 기법은 룰렛 휠 기법이다 [Noraini, M. R. 등, 2001]. 룰렛 휠은 적합도 점수가 높을수록 룰렛에서 선택될 확률을 높여주는 기법이다. 그림 5는 적합도 점수가 가장 낮은 부모(P1)부터 가

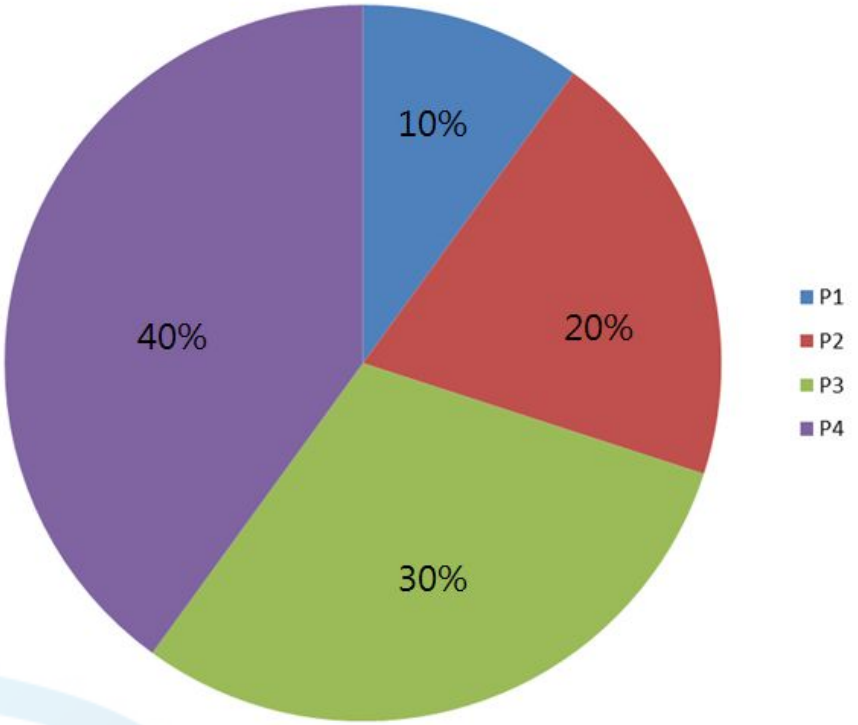



그림 5. 룰렛 휠 부모선택 기법

장 높은 부모(P4)를 룰렛 휠 위에 설정하였을 때를 보여준다. P1의 경우 적합도 점수가 가장 낮기 때문에 룰렛의 10%의 공간만을 차지하고 있지만 P4의 경우 점수가 높기 때문에 가장 넓은 40%의 공간을 차지하고 있다.

두 부모가 선택되고 나면 두 부모로부터 두 자녀가 생성되는 과정인 교배가 이루어진다. 교배를 통해 생성되는 두 자녀는 두 부모로부터 유전자를 상속 받는다. 상속 받는 과정 속에서 유전자의 교차와 변이가 이루어진다[Wright, S., 1932]. 유전자의 교차는 두 부모의 이진 비트열을 갖는 유전자 중 특정한 위치를 기준으로 하여 두 유전자의 정보가 서로 교차하여 합쳐지는 것을 의미한다. 그리고 유전자의 변이는 염색체가 갖는 유전자의 비트열 중 하나의 비트의 값이 변화하는 것을 의미하며 만약 1이었다면 0으로, 0이었다면 1로 변화하게 된다. 변이는 특정한 확률에 의해 발생하게 되는데 사용자가 유전자 알고리즘을 시작하기 전 파라미터 값으로서 설정할 수 있다. 그림 6은 두 부모 P1과 P2가 교배하여 자녀를 생성할 때 두 부모의 유전자의 교차와 변이가 이루어 지는 것일 보여주고 있다. 그림 6(a)는 P1과 P2의 유전자를 보여주고 있고 회색 바탕의 유전자는 P1의 유

전정보를 나타내며, 흰색 바탕의 유전자는 P2의 유전정보를 나타낸다. 그림 6(b)는 두 유전자의 교차가 이루어지는 것을 보여주고 있다. 두 유전자의 교차 지점인 6비트열의 정 가운데를 기준으로 각 부모의 3비트열이 서로 교차함을 볼 수 있다. 그림 6(c)는 교차한 후 유전자의 변이를 보여주고 있다. 붉은색 바탕의 비

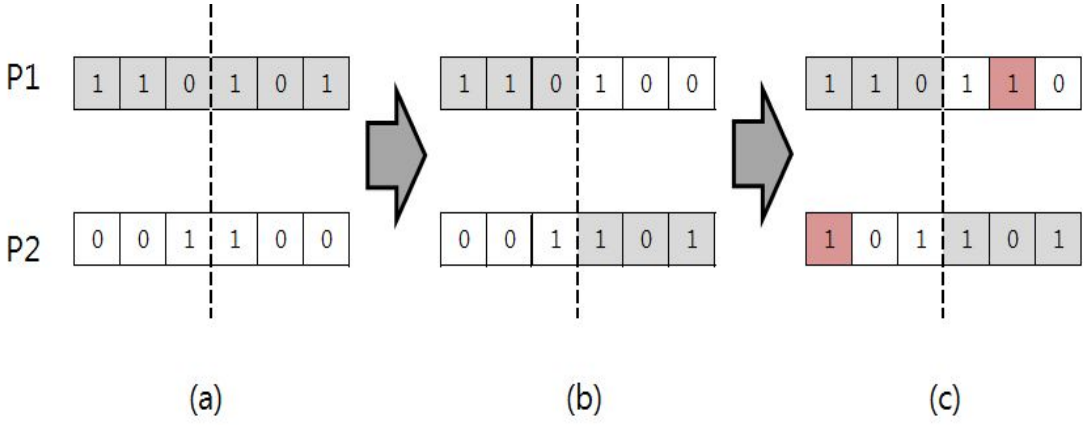



그림 6. 유전자 알고리즘의 교배 과정 (a) 두 부모 P1, P2의 유전자
(b) 유전자의 교차 (c) 유전자의 변이

트를 보면 그림 6(b)에서 0이었던 경우 변이를 통해 1로 변하였음을 볼 수 있다. 그림 6과 같은 과정을 통해 두 자녀가 생성되는데 이 때 각 자녀가 갖는 유전자는 그림 6(c)의 두 유전자이다. 두 부모의 교배 과정에서 발생하는 교차와 변이를 통해 생성된 자녀 염색체는 부모와는 다른 적합도 점수를 가질 수 있다. 만약 적합도가 높아질 경우 다음 세대의 부모로 선택될 가능성이 있지만 반대로 적합도가 낮아질 경우 다음 세대에 부모로 선택될 가능성이 낮아진다.

염색체의 적합도 평가, 부모의 선택, 두 부모의 교배를 통한 유전자의 교차와 변이가 매 세대마다 이루어지며 세대가 지날수록 그 세대에 존재하는 염색체들이 갖는 유전자는 점점 문제에 대해 최적화 되어 간다. 하지만 시뮬레이션 결과는 유전자의 교차율과 변이율에 대해 민감하게 반응한다[Lin, W. Y. 등, 2003]. 만약 유전자의 변이가 너무 자주 일어나게 될 경우 매 세대마다 평가도가 들쭉날쭉하게 되어 시뮬레이션이 끝날 때까지 결과가 수렴하지 못 할 수도 있다. 반대로 변이율이 너무 낮다면 다른 해답을 찾지 못하고 결과가 조기 수렴할 수 있다.

유전자 알고리즘의 종료 조건은 설정된 최대 세대가 끝날 때 혹은 어느 정도 세대가 지났음에도 불구하고 염색체들의 평균 적합도 점수가 특정 비율 이상으로

증가하지 않을 경우이다. 적합도 점수가 특정 비율 이상으로 증가하지 않을 경우는 크게 두 경우가 있는데 첫 째는 초기에 생성된 염색체들의 유전자가 매우 뛰어났거나 혹은 조기 수렴할 경우이다. 조기 수렴이란 유전자 알고리즘이 현재 찾은 해답보다 더 최적화된 해답이 있음에도 불구하고 더 이상 찾기 못하고 종료하는 경우이다. 이러한 현상이 발생하는 이유는 두 부모의 교배 과정의 교차와 변이의 확률 때문이다. 교차와 변이의 확률이 너무 낮아서 자녀들의 유전자 적합도가 부모의 적합도와 별로 차이가 없게 되는 경우가 지속적으로 발생할 경우 세대간의 적합도 증가율이 작아지기 때문에 종료조건에 의해 시뮬레이션이 종료되게 된다. 그렇기 때문에 아직 최적화된 답을 찾지 못했음에도 불구하고 그 해답이 결과로 나타나게 된다. 반면에 교차와 변이율이 너무 높아서도 안 된다. 이 같은 경우 각 세대마다 적합도 점수가 들쭉날쭉 할 수 있기 때문에 유전자 알고리즘을 적용할 경우 이러한 요소를 잘 조절해 주어야 한다.

# III. 셀 우주: 셀 수준 진화 프레임워크

## 3.1 구조

셀 우주(CC: Cellular Cosmos)란 특정한 행위결정 로직프레임을 갖는 인공 개체가 2차원의 유한한 셀 공간상에서 자신의 이웃 셀에 존재하는 먹이에 대해 인식하여 세대가 지남에 따라 환경에 적합한 형태로 진화하는 셀 수준의 진화 시뮬레이션 프레임워크이다[정보선, 정성훈., 2014 : 정보선, 정성훈., 2015]. 세포 자동자에서의 셀들은 유한한 상태를 갖는 진화의 대상이지만 셀 우주에서의 셀들은 하나 혹은 다수의 인공개체와 먹이가 존재할 수 있는 공간이다. 이 인공개체들은 행위결정 로직프레임의 내부로직을 통하여 출력을 내보내게 되고 그 출력값에 따라 특정한 행동을 하게 된다. 그리고 이 공간상에 존재하는 인공개체의 행동을 결정짓는 내부로직이  셀 우주의 진화 대상이다. 초기에는 인공개체들이 임의의 내부로직을 갖기 때문에 먹이를 잘 먹지 못하지만 시간이 지날수록 먹이를 잘 먹는 방향으로 진화하게 된다. 셀 우주의 시간은 세포 자동자와 마찬가지로 동기식이진 시간[Ossimitz, G. 등, 2008]을 갖는다. 세포 자동자는 각 시간단계가 하나의 세대이지만 셀 우주에서는 시간개념이 상위단계와 하위단계로 나뉘게 된다. 하위 시간단계가 하나씩 증가를 하며 일정한 수치에 도달하게 되면 0으로 리셋되며 상위 시간단계 1개가 증가하게 된다. 각 개체들은 하위 시간단계가 1씩 증가할 때 마다 동시에 행동한다.

상위 시간단계가 1이 증가할 때 마다 개체들은 생식을 통해 자녀개체를 생성한다. 이 때 유전자 알고리즘의 경우 두 부모를 선택하여 두 자녀를 생성하였지만 셀 우주의 경우 하나의 부모가 하나의 자녀를 생성하는 자기번식을 한다. 이 때 생성되는 자녀 개체의 유전 정보는 부모의 유전정보를 상속받으며 확률적으로 변이가 일어난다. 개체들은 단성생식을 하기 때문에 유전정보의 교차는 없으며 오로지 변이[Libelli, S. M. 등, 2000]를 통해 새로운 행동을 하는 유전자를 만들게 된다. 이 과정이 반복되며 환경에 적합한 행동을 하는 개체들은 살아남게 되고 반대로 적합하지 못한 행동을 하는 개체들은 도태하여 살아남지 못하게 되며 진

화가 이루어진다[Darwin, C., 2009].

## 3.2 시뮬레이션 환경

셀 우주는 그림 7과 같은 2차원의 유한한 셀 격자 공간을 가지고 있으며 이 공간상에 인공개체(삼각형)와 먹이(십자형)가 존재한다. 처음 시뮬레이션 시작 시 임의의 개수만큼의 인공개체와 먹이를 생성한다. 인공개체는 셀 공간상에서 자신의 이웃하는 셀 환경에 먹이가 존재하는 것을 인식하고 이에 대해 행동을 하게 된다. 행동을 하기 위하여 모든 인공개체는 동일한 구조를 갖는 행위결정 로직프레임을 가지고 태어난다. 그리고 행위결정 로직프레임이 갖는 내부로직에 의하여 인공개체의 행동이 결정된다. 이 내부로직은 인공개체들이 태어날 때 임의적으로


그림 7. 셀 우주의 2차원 셀 격자 공간

결정되거나 부모로부터 물려받아 유전적으로 결정된다. 인공개체들이 공통적인 행위결정 로직프레임을 갖고 있지만, 서로 다른 내부로직을 가질 수 있기 때문에 인공개체들은 서로 다른 행동을 할 수 있게 된다. 행위결정 로직프레임은 인공개체가 인식할 수 있는 5개의 이웃 셀 환경을 입력으로 받아 내부로직으로 처리하여 행동을 출력한다. 인공개체는 행위결정 로직프레임을 갖는 반면 먹이들은 갖지 않는다. 그렇기 때문에 먹이들은 아무런 행동을 하지 않고 가만히 있게 된다. 다만 죽은 먹이는 일정시간 지나면 재생성 되는데 이 때 처음 생성된 위치에서


그림 8. 인공개체의 이웃 셀

혹은 임의의 위치에서 다시 생성된다.

행위결정 로직프레임은 인공개체가 인식하는 이웃 셀의 상황을 비트단위로 입력 받는다. 인공개체가 인식하는 이웃 셀은 그림 8번의 N1~N5와 같다. 그림 8는 북쪽으로 이동하고 있는 인공개체를 보여주고 있는데 이 때 인공개체가 인식할 수 있는 이웃 셀은 그림과 같이 진행방향을 기준으로 인공개체의 앞 쪽의 5개의 셀이다. 인공개체는 이웃 셀에 먹이가 존재하는지 확인을 하며 만약 먹이가 존재할 시 1의 값을, 존재하지 않을 시 0의 값을 행위결정 로직프레임의 입력으로 주게 된다. 그렇기 때문에 만약 N1에 먹이가 존재한다면 N1의 입력값은 1이 된

다. 행위결정 로직프레임은 N1~N5의 5비트 입력값을 받은 후 내부로직을 통해 출력값을 내보낸다. 출력값은 2비트로 구성되어 있으며 이에 대해 결정된 행위는 그림 표 1과 같다. 표 1의 O1은 앞의 비트를, O2는 뒤의 비트를 보여준다. 만약 출력값이 (0,0)이면 인공개체는 아무런 행동을 하지 않고, (0,1)이면 오른쪽으로 방향전환한 후 한 칸 이동, (1,0)이면 왼쪽으로 방향전환 후 한 칸 이동, 그리고 (1,1)일 경우 현재 진행방향을 기준으로 직진을 하게 된다. 만약 시간 t-1일 때 멈춰있던 인공개체가 t일 때 어떠한 행동을 한다면, 이 때의 방향은 가장 최근에 진행하였던 방향을 기준으로 한다.

표 1. 인공개체의 출력에 대한 행동

| O1 | O2 | 행동 |
|---|---|---|
| 0 | 0 | 멈춤 |
| 0 | 1 | 우회전 |
| 1 | 0 | 좌회전 |
| 1 | 1 | 직진 |

셀 공간상에서 인공개체가 움직일 수 있는 방향은 동, 서, 남, 북 네 방향이다. 그리고 인공개체들이 현재 움직이고 있는 방향을 셀 공간상에서 삼각형의 꼭지점을 통해 확인할 수 있다. 삼각형 하나의 꼭지점이 동, 서, 남 혹은 북 쪽을 가리킬 경우 이 인공개체는 꼭지점의 방향으로 이동하고 있음을 알 수 있다.

셀 우주의 시간은 상위 시간단위인 '세대'와 하위 시간단위인 '단위시간'으로 나뉘며 동기화 이산시간을 가진다. 하위단위인 단위시간은 0부터 시작하여 이산시간이 1개씩 증가할 때 마다 +1씩 수행하게 된다. 단위시간이 일정한 수치만큼 도달하게 되면 단위시간은 0으로 초기화 되며 세대는 +1이 된다. 셀 우주에서 인공개체들이 자신의 셀 이웃 상황을 파악하고 내부로직을 통해 행동하기 까지 걸리는 시간은 1단위시간이다. 각 인공개체가 살아서 자녀를 생성하거나 죽게 되는데 걸리는 시간은 1세대이다. 인공개체들은 한 세대동안 살며 적어도 하나의 먹이를 먹으면 다음세대까지 살아남아 자녀를 생성하게 되지만 하나도 못 먹은 경우 인공개체는 살아남지 못하고 죽게 된다. 인공개체가 먹이를 1개 이상 먹어도 죽는 경우가 있는데 이 경우는 인공개체가 포화상태에 이르렀을 때이다. 시뮬레이션 후반으로 갈수록 인공개체들은 먹이를 많이 먹기 때문에 인공개체가 죽지

않고 포화상태에 도달한다. 이 때 인공개체들이 죽는 조건은 한 세대동안 모든 인공개체들이 먹은 먹이의 수를 평균 내어 평균 미만의 먹이를 먹은 인공개체들은 살아남지 못하고 죽게 된다.

시뮬레이션 후반으로 갈수록 인공개체가 먹이를 잘 먹는 이유는 다음과 같다. 초기에는 임의의 내부로직을 갖는 인공개체들이 태어나기 때문에 먹이를 잘 먹지 못한다. 하지만 동일한 세대에 살아가는 인공개체들 중 비교적 먹이를 잘 먹는 개체들은 살아남아 번식하게 되는데 이 때 자녀에게 자신의 변이된 내부로직을 상속시켜 주게 된다. 변이된 내부로직을 상속받은 자녀들은 부모와 다른 행동을 보일 수 있게 된다. 만약 이 행동이 먹이를 잘 먹는 행동이라면 이 개체는 잘 살아남게 된다. 이 과정이 세대마다 반복되기 때문에 세대가 흐를수록 인공개체들은 먹이를 잘 먹게 된다.

우리는 셀 우주의 시뮬레이션 환경을 위에서 설명한 것과 같이 인공개체-먹이 모델에 적용하여 두 종류의 인공개체에 대해 실험을 하였다. 첫 번째 인공개체는 기억회로가 없는 인공개체에 대한 실험이고 두 번째 인공개체는 기억회로가 있는 인공개체에 대한 실험이었다. 실험을 통해 두 인공개체의 행동을 확인해 보았고 또한 진화적으로 왜 인공개체들이 그렇게 행동하는지 분석하여 보았다.

## 3.3 인공개체-먹이 모델 적용 시뮬레이션

본 논문에서는 셀 우주의 인공개체-먹이 모델에서 인공개체가 기억회로를 갖지 않을 때와 인공개체가 기억회로를 가질 때 두 종류로 나누어 실험해 보았다. 인공개체가 갖는 기억회로는 행위결정 로직프레임에 해당하는 요소이다. 기억회로를 갖지 않는 인공개체의 경우 행위결정 로직프레임은 오로지 이웃 셀의 상황에 대하여 입력을 받고 내부로직을 통해 출력값을 내보낸다. 그렇기 때문에 기억회로가 없는 인공개체가 만약 동일한 먹이상황에 놓여 있다면 항상 동일한 행동을 반복 수행할 것이다. 하지만 기억회로가 있는 행위결정 로직프레임은 이웃 셀의 상황을 입력으로 받음과 동시에 인공개체가 과거에 했던 자신의 행동을 되먹임 받아 현재의 행동을 출력한다. 그렇기 때문에 기억회로를 갖는 인공개체는 동일한 먹이상황에 놓여있더라도 과거의 행동이 무엇이었는지에 따라 다른 행동을

할 수 있게 된다. 이번 절에서는 셀 우주의 인공개체-먹이 모델에서 행위결정 로직프레임에 기억회로가 없을 때와 있을 때 어떻게 구성되어 있는지 그리고 어떻게 실험하였는지를 기술한다.

### 3.3.1 기억회로가 없는 경우

셀 우주의 인공개체-먹이 모델의 첫 번째 실험은 기억회로가 없는 인공개체에 대한 실험이다. 그림 9는 기억회로를 갖지 않는 인공개체의 행위결정 로직프레임이다. N1~N5는 이웃 셀의 먹이상황에 대한 입력을 보여주고 있다. 만약 N1에 먹이가 있을 경우 N1은 1이 입력되고 빈 공간일 경우 0이 입력된다. 각 먹이에 대한 입력은 정상입력과 보수입력으로 나뉘어 총 8개의 AND게이트(A1~A8)에 입력된다. 하지만 먹이에 대한 입력과 AND게이트 사이의 연결되는 교차점들에는 각각 접점이 존재할 수 있다. 만약 교차점에 접점이 없다면 이 교차점에 해당하는 먹이입력은 AND게이트의 입력으로 들어오지 않으며 접점이 존재할 때에만 입력으로 들어온다. 그렇기 때문에 교차점의 접점 존재 여부에 따라 먹이입력 N1~N5가 AND게이트의 입력으로서 들어오는지를 결정하게 된다. 만약 N1과 A1에 접점이 있고 N3와 A1에 접점이 있을 경우 A1의 출력값은 N1*N3가 된다. 이처럼 나머지 8개의 AND게이트들 또한 출력을 내보내게 되고 이 출력값들은 각각 2개의 OR(O1, O2)게이트에 입력으로 들어간다. O1의 경우 A1~A4의 출력이 입력으로 들어오고 O2의 경우 A5~A8의 출력이 입력으로 들어온다. 위와 같은 방법으로 O1과 O2의 출력이 결정되고 출력에 따라 인공개체의 행동이 결정된다. O1과 O2는 각각 표 1의 출력비트이며 인공개체의 출력 별 행동 또한 표 1과 같다.

행위결정 로직프레임상의 접점의 위치와 개수에 의해 3.2절에서 설명한 내부로직이 결정된다. 또한 인공개체가 자녀를 생성할 때 유전해주는 내부로직이 부모의 접점의 개수와 그 위치이다. 그리고 유전 받는 과정 속에서 이루어  지는 내부로직의 변이는 접점의 위치와 개수에 대한 변화이다. 행위결정 로직프레임의 입력과 AND게이트 사이의 교차점에서 각각 확률적으로 변이가 일어나는데 만약 접점이 있는 교차점에서 변이가 발생했을 때 그 접점은 없어지고 반대로 접점이

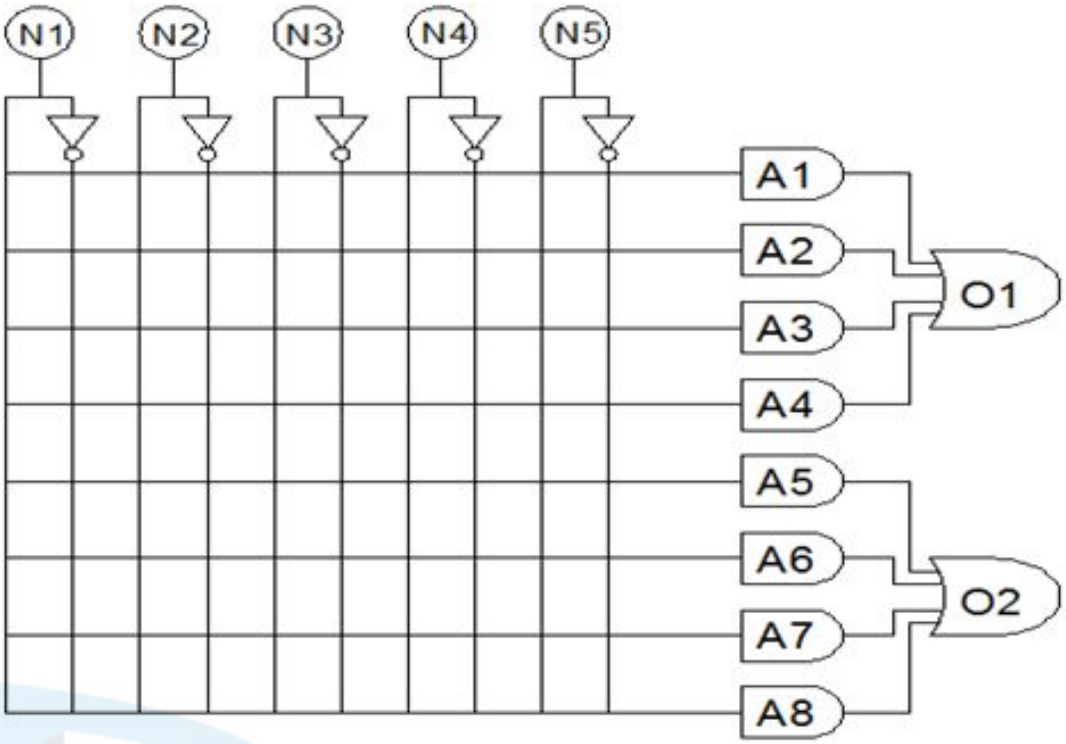



그림 9. 기억회로가 없는 행위결정 로직프레임

없는 교차점에서 변이가 발생했을 때 그 교차점에 접점이 생성된다. 이러한 변이 과정을 통해 접점의 위치와 개수가 변하여 인공개체가 갖는 내부로직이 변하며 행동 또한 변하게 된다. 그림 10은 단순한 행위결정 로직프레임에서 N3와 A2의 교차점에 변이가 발생했을 때 내부로직이 어떻게 변하는지 보여주고 있다. 변이 전인 그림 10(a)는 N1,N2와 A1,A2의 교차점에 접점(붉은 점)이 존재하고 있다. 이 경우 A1의 출력은 N1*N2이고 A2또한 N1*N2이기 때문에 내부로직은 두 AND게이트의 OR연산에 대한 결과 N1*N2+N1*N2가 된다. 이러한 내부로직은 갖는 행위결정 로직프레임에 N3와 A2의 교차점에 변이가 발생하여 그림 10(b)와 같이 되었다면 변이된 내부로직은 N1N2+N1N2N3가 된다. 행위결정 로직프레임의 특성상 AND게이트에 오로지 하나의 접점만 존재하여 하나의 입력만이 들어오는 경우가 발생할 수 있는데 이와 같은 경우에는 입력 값이 그대로 출력된다.

셀 우주에서 각 셀에 인공개체 또는 먹이가 존재할 수 있다. 인공개체의 행동은 진화의 대상이고 먹이는 인공개체가 진화하기 위해 필요한 대상이다. 그렇기 때문에 인공개체는 먹이를 먹어야 하는데 이 인공개체가 먹이가 존재하는 셀 공

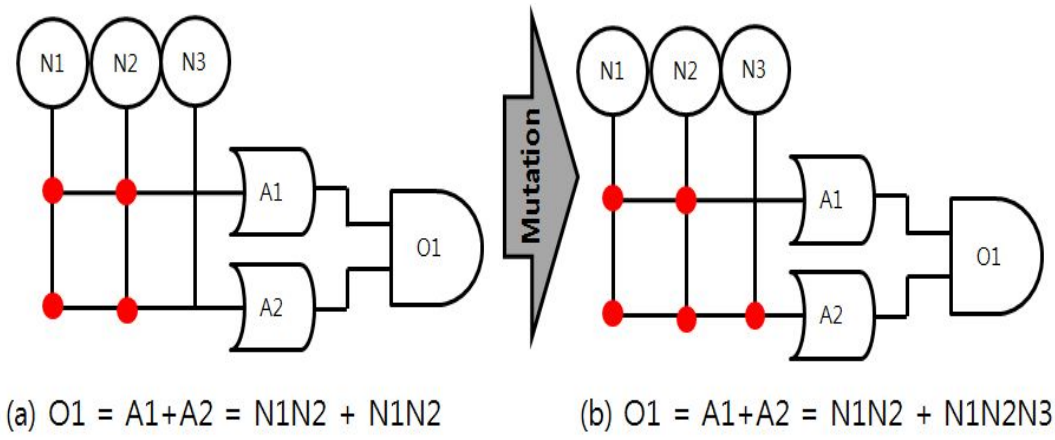



그림 10. 내부로직 변이 (a) 변이 전 (b) 변이 후

간으로 이동하였을 경우 이 먹이를 먹었다고 인식하며 먹이는 죽게 된다. 죽은 먹이가 재 생성되는 조건은 죽고 난 후 일정 단위시간 후 이다. 먹이가 다시 태어나는 위치는 임의의 빈 셀 공간으로 설정하거나 혹은 죽었던 위치에서 다시 태어나도록 설정할 수 있다.

기억회로가 없는 행위결정 로직프레임에서 인공개체가 가질 수 있는 이웃 셀의 상황은 한 셀당 먹이가 있거나 없는 상황이므로 1비트로 표현 가능하며 모두 5개의 이웃 셀을 인식하므로 총 5비트의 입력이 되며 이는 32가지의 입력 경우의 수를 가진다. 만약 인공개체가 32개의 경우에 대해 모두 먹이를 먹는 쪽으로 행동을 한다면 가장 좋은 내부로직을 갖고 있다고 할 수 있다. 우리는 설계한 행위결정 로직프레임이 32점을 갖는 내부로직을 가질 수 있는지 검증하기 위하여 몬테카를로 방법[MacKay, D. J., 1998]을 사용하였다. 몬테카를로 방법을 통해 첫 번째 실험인 출력 O1과 O2에 대한 인공개체의 행동을 결정했으며 이를 통해 표 1의 결과를 얻을 수 있었다. 또한 이 방법을 통해 두 번째 실험인 행위결정 로직프레임이 먹이를 잘 먹을 수 있는 내부로직을 가질 수 있는지 확인해 보았다.

첫 번째 실험을 위하여 내부로직을 임의적으로 형성하여 32개의 입력을 주었을 때 나오는 출력에 대한 경우의 빈도수를 파악하였다. 임의의 내부로직을 형성하기 위하여 행위결정 로직프레임의 각 교차점에 접점이 생길 확률을 0.1로 두어 임의의 내부로직을 형성하였다. 그리고 이 내부로직에 32개의 입력을 순차적으로 주었을 때 2비트의 출력(O1, O2)이 어떻게 나오는 지 확인하여 카운팅 하였다.

이 과정을 총 1000번 반복하여 출력의 경우의 수를 합산하였다. 그 결과 표 2와 같은 모습을 볼 수 있었다.

표 2. 몬테카를로 방법을 통한 인공개체의 행동결정

| O1 | O2 | Count | Ratio[%] |
|---|---|---|---|
| 0 | 0 | 3478 | 10.9 |
| 0 | 1 | 7215 | 22.5 |
| 1 | 0 | 7125 | 22.3 |
| 1 | 1 | 14182 | 44.3 |

O1과 O2는 행위결정 로직프레임의 출력비트이며 count은 1000번 씩 32개의 입력에 대해 해당 출력비트가 나왔을 때를 카운팅 한 것이다. 그리고 ratio는 해당 출력비트가 나온 빈도를 백분율로 나타낸 것이며 이를 구하는 수식인 ratio = (100*count)/(32*1000)를 적용하여 계산하였다. 그 결과 행위결정 로직프레임이 내보내는 출력(O1,O2)이 (0,0)인 경우 10.9%로 빈도수가 가장 낮았고 (0,1) 혹은 (1,0)인 경우가 22.5%와 22.3%로 거의 비슷하였으며 (1,1)인 경우가 44.3%로 가장 높았다. 인공개체의 내부로직이 좋지 못하더라도 적어도 하나의 먹이를 먹을 가능성이 가장 큰 경우는 무조건 직진할 때 이다. 그렇기 때문에 환경에 가장 잘 적응할 수 있는 행동을 빈도수가 가장 높은 경우로 설정하였으며 이 경우 (1,1)의 출력이 직진이 된다. 또한 우회전이나 좌회전의 경우 비슷한 행동이라 판단되어 우회전을 (0,1), 좌회전을 (1,0)으로 설정하였다. 그리고 가장 좋지 못한 행동인 멈춤을 빈도수가 가장 낮은 (0,0)으로 설정하였다.

몬테카를로 방법이 적용된 두 번째 실험은 32점의 내부로직을 찾는 실험이었다. 이 실험을 위하여 위와 동일하게 임의의 내부로직을 형성하였으며 각 32개의 입력에 대하여 내부로직이 먹이를 먹는 방향으로 이동하면 +1점을, 그렇지 아니하면 +0점을 주었다. 이번 실험은 총 120,000번 하였다. 이렇게 많이 실험한 이유는 1,000번 해 보았을 때 32점짜리 로직은 찾지 못하였고 최대 28점을 찾았기 때문에 더 많은 반복실험이 필요하다고 판단하였기 때문이다. 그 결과 32점을 찾기는 하였지만 빈도수가 매우 낮아 크게 의미가 없다고 판단하여 처음 결과인 28점 이상의 내부로직이 얼마나 많이 발생하는지 확인해 보았으며 총 120,000번 중 110번인 약 0.1% 발생하였다. 하지만 실험의 목적은 행위결정 로직프레임의 구조가 32점을 가질 수 있는지를 보는 실험이었으며 32점짜리 내부로직을 가질 수

있음을 확인하였다.

몬테카를로 방법을 통해 얻은 결과를 기반으로 우리는 행위결정 로직프레임이 32점 내부로직을 가질 수 있음을 확인하였다. 그 다음으로는 실험환경의 검증이었다. 실험환경을 검증하는 실험을 하기 위하여 우리는 적합도 평가[Fan, W. 등, 2004]를 도입하였다. 각 시뮬레이션의 세대가 끝날 때 마다 모든 인공개체의 로직을 32개의 입력에 대해 어떠한 행동을 하는지 확인 후 적합도 점수를 주었다. 또한 적합도가 높은 개체가 죽지 않도록 Elitism을 적용하였다[Yang, S., 2007]. 이 실험의 결과는 5절에서 설명하도록 하겠다.

위의 두 검증 방법을 통해 우리는 로직프레임의 구조와 시뮬레이션 환경이 적용 가능하다고 판단하였다. 이를 근거로 우리는 여러 먹이 구조를 통해 인공개체가 어떻게 진화하는지 실험해 보았으며 그 결과 간단한 먹이구조에서 인공개체가 먹이를 잘 찾아 먹도록 진화한 것을 확인하였다.

### 3.3.2 기억회로가 있는 경우

셀 우주에서 기억회로를 갖는 인공개체의 행위결정 로직프레임의 구조는 그림 11과 같다. 기억회로가 없는 행위결정 로직프레임과 전체적인 구조는 동일 하지만 자신의 행동을 기억하여 입력으로 받는 회로를 추가로 가지고 있다. 기억회로는 인공개체가 과거에 했던 행동의 2비트 값을 기억하였다가 현재의 입력으로 되먹임 받는 구조를 가지고 있다. 인공개체가 과거행동을 기억할 수 있는 시간단계는 설정 값으로서 사용자가 원하는 만큼 설정할 수 있다. 그렇기 때문에 실험하고자 하는 먹이구조나 환경에 대해 적절한 행동을 기억하도록 설정하여야 한다.

그림 11의 점선 사각형 내부는 추가된 기억회로 부분이며 4개의 기억단계까지 행동을 기억할 수 있는 것을 보여주고 있다. 그리고 이 부분을 제외한 나머지는 일반적인 행위결정 로직프레임과 동일함을 확인할 수 있다. 기억회로 부분의 동작원리는 다음과 같다. 만약에 A라는 인공개체가 T시간단계에서 P출력 내보내어 행동을 한 후 T+1단위시간으로 넘어간 상황에 놓여있다고 가정 했을 때 T단위시간에서의 P는 $Z^{-1}$블록을 지나게 되고 기억에 대한 입력 중 한 부분인 t−1입력으로 들어가게 된다. 그리고 T단위시간일 때의 t−1에 놓여있던 입력은 왼

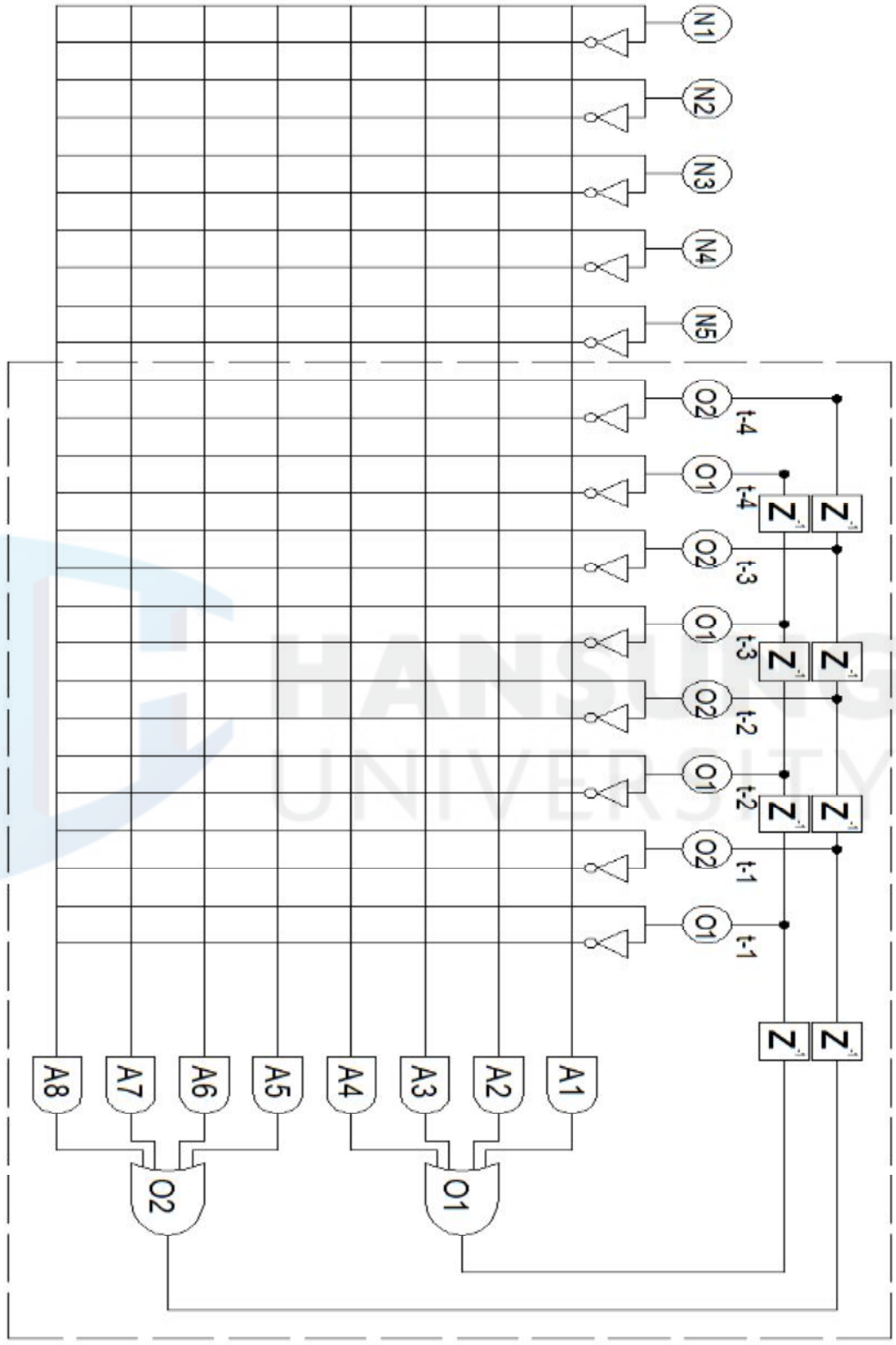



그림 11. 기억회로를 갖는 행위결정 로직프레임

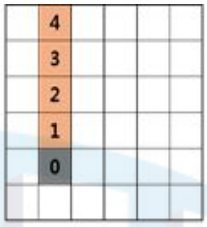
쪽에 존재하는 $Z^{-1}$블록을 지나며 t-2입력으로 들어가게 된다. 나머지 t-3, t-4입력에 대해서도 동일하게 동작하며 가장 오래된 기억을 할당받는 t-4에 있었던 입력은 사라진다. 이와 같은 과정을 통해 인공개체는 자신의 과거 행동을 기억하며 현재의 입력으로 되먹임 받아 현재의 행동을 결정하게 된다. 그렇기 때문에 인공개체의 먹이상황이 과거와 동일하더라도 기억회로가 있는 경우 과거에 어떠한 행동을 했는지에 따라 다른 행동을 할 수 있다.

기억회로를 갖는 인공개체는 기억회로가 없는 인공개체보다 복잡한 먹이구조에서 더 좋은 행동을 할 수 있다. 기억회로가 없는 인공개체의 경우 오로지 먹이상황을 입력으로 받아 현재의 행동을 출력하기 때문에 먹이구조가 조금이라도 복잡해지면 먹이를 잘 먹지 못하게 된다. 하지만 기억회로가 있는 인공개체는 먹이상황 외에 자신의 과거 행동을 입력으로 받아 행동을 결정하기 때문에 더욱 다양한 행동을 할 수 있게 되며 이 행동을 통하여 복잡한 먹이상황에서도 먹이를 잘 먹도록 행동을 하게 된다. 예를 들어 아무런 먹이가 없는 상황에서 놓여있는 인공개체가 갖는 먹이에 대한 입력 N1~N5는 모두 0이다. 그렇기 때문에 기억회로가 없는 인공개체의 경우 만약 직진을 했다면 항상 직진을 할 것이고, 우회전을 했다면 항상 우회전을 할 것이다. 하지만 기억회로가 있는 개체의 경우 T단위시간에서 직진을 했다면 T+1단위시간에서는 (1,1)을 t-1기억입력으로 받을 것이다. 그렇기 때문에 T시간단계의 t-1기억입력이 (1,1)이 아니었다면 이 인공개체의 기억입력이 바뀌게 되며 먹이조건이 동일하더라도 시간에 따라 다른 행동을 할 수 있게 된다.

그림 12는 먹이가 없는 상황에서 기억회로가 없는 인공개체와 1단위시간까지 기억하는 인공개체가 할 수 있는 행동에 대한 예를 보여주고 있다. 그림 12(a)의 경우 기억회로가 없는 인공개체가 회색의 0번 셀에서 출발하여 순차적으로 4번 셀 까지 가는 경로를 붉은색으로 보여주고 있다. 주변에 아무런 먹이가 없는 상황이기 때문에 행위결정 로직프레임의 먹이에 대한 입력 N1~N5는 모두 0으로 동일함을 볼 수 있다. 이 경우 단위시간 T에서 출력인 O1(t)와 O2(t)는 1과 1로서 이 인공개체는 직진을 하게 된다. 직진을 하여 1번으로 이동하였지만 아무런 먹이가 없기 때문에 이전 상태와 동일하여 또다시 직진을 하게 된다. 이 인공개체는 먹이를 찾지 못한다면 항상 북쪽으로만 이동할 것이다. 그림 12(b)는 1개의

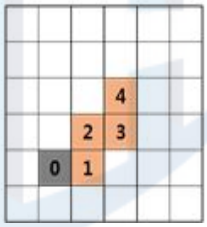
단위시간까지의 행동을 기억하는 회로를 갖는 인공개체가 0번의 셀에서 출발하여 4번 셀까지 가는 경로를 보여주고 있다. 그림 12(a)와는 다르게 인공개체가 항상 동일한 행동을 하지 않고 우회전, 좌회전을 반복하며 이동하고 있음을 볼 수 있다. 인공개체가 갖는 먹이입력은 그림 12(a)와 동일하다. 하지만 1개의 단위시간 전의 행동을 기억하여 입력으로 받는 부분인 O1(t−1)과 O2(t−2)가 변함을 볼 수 있다. 그림 12(b)의 표와 같이 T단위시간 일 때 출력인 O1(t)와 O2(t)가 T+1에서의 기억입력인 O1(t−1)과 O2(t−2)로 입력받아 T와 T+1의 행동이 변화함을 볼 수 있다. 마찬가지로 T+1과 T+2일 때에도 동일한 현상이 일어나며 인공개체가 좌회전, 우회전을 반복하여 행동함을 볼 수 있다.

(a)




| time | path | input | | | | | output | | |
|---|---|---|---|---|---|---|---|---|---|
| | | N1 | N2 | N3 | N4 | N5 | O1(t) | O2(t) | action |
| T | 0 -> 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | forward |
| T+1 | 1 -> 2 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | forward |
| T+2 | 2 -> 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | forward |
| T+3 | 3 -> 4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | forward |

(b)




| time | path | input | | | | | output | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | N1 | N2 | N3 | N4 | N5 | O1(t-1) | O2(t-1) | O1(t) | O2(t) | action |
| T | 0 -> 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | right |
| T+1 | 1 -> 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | left |
| T+2 | 2 -> 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | right |
| T+3 | 3 -> 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | left |

그림 12. 기억회로가 없는 개체와 있는 개체의 행동 비교

우리는 두 인공개체의 행동을 다양한 먹이구조에서 실험하였을 때 개체들이 어떻게 행동을 하였는지, 어떤 개체가 더 먹이를 잘 먹었는지, 그리고 그 결과에 대해 실험하고 분석하였다.

# IV. 연 구 결 과

## 4.1 기억회로가 없는 경우

셀 우주의 기억회로가 없는 인공개체-먹이 모델에서는 주요 두 실험을 하였다. 첫 번째로 실험환경과 행위결정 로직프레임의 동작을 확인하기 위하여 인공개체들의 적합도를 통한 진화를 실험하였다. 두 번째로는 단순히 먹이를 먹고 생식과 변이를 통한 진화를 다양한 먹이 패턴에서 실험하였다.

### 4.1.1 적합도 평가를 통한 인공개체의 진화

적합도란 개체가 어떠한 환경에 얼마나 최적화되어 있는지 평가하는 것이다. 인공개체의 적합도를 평가하기 위하여 행위결정 로직프레임에 들어올 수 있는 모든 입력에 대하여 가능한 행동들 중 먹이를 먹는 행동들이 무엇인지 분별해 낼 필요가 있었다. 분멸해 내기 위하여 가능한 행동들 중 가장 최적의 행동이 무엇인지 찾아보았다. 그 결과 표 3과 같은 결과를 얻었다. 표 3은 32개의 모든 입력에 대하여 인공개체의 최적화된 행동이 무엇인지를 case1~3을 통해 보여주고 있다. 인공개체는 특정한 먹이상황에 대하여 최적화된 행동을 동시에 3개까지 가질 수 있기 때문에 case를 3개로 나누었으며 각 케이스의 우선순위는 모두 동일하다. 예를 들어 N2, N4, N5에 먹이가 존재하는 인공개체의 경우 직진이나 좌회전, 혹은 우회전을 하더라도 먹이를 먹을 수 있다. 그렇기 때문에 3개의 case에 대해 어떠한 행동을 해도 무관하다.

위의 예는 인공개체가 1단위시간 안에 먹이를 먹을 수 있는 경우이다. 하지만 항상 인공개체가 1단위시간 만에 먹이를 먹을 수 있지 않다. 예를 들어 N1위치에 먹이가 있는 경우 이 인공개체는 먹이를 먹기 위해 적어도 2단위시간이 소요된다. 그렇기 때문에 먹이의 상황에 따라 최적화된 행동이 바뀔 수 있으며 각 상황은 아래의 3개로 나뉜다.

| 입력 | | | | | | 행동 | | |
|---|---|---|---|---|---|---|---|---|
| Number | N1 | N2 | N3 | N4 | N5 | Case 1 | Case 2 | Case 3 |
| 1 | 0 | 0 | 0 | 0 | 0 | 직진 | X | X |
| 2 | 1 | 0 | 0 | 0 | 0 | 직진 | 좌회전 | X |
| 3 | 0 | 1 | 0 | 0 | 0 | 직진 | X | X |
| 4 | 1 | 1 | 0 | 0 | 0 | 직진 | X | X |
| 5 | 0 | 0 | 1 | 0 | 0 | 직진 | 우회전 | X |
| 6 | 1 | 0 | 1 | 0 | 0 | 직진 | 좌회전 | 우회전 |
| 7 | 0 | 1 | 1 | 0 | 0 | 직진 | X | X |
| 8 | 1 | 1 | 1 | 0 | 0 | 직진 | X | 우회전 |
| 9 | 0 | 0 | 0 | 1 | 0 | 좌회전 | X | X |
| 10 | 1 | 0 | 0 | 1 | 0 | 좌회전 | X | X |
| 11 | 0 | 1 | 0 | 1 | 0 | 직진 | 좌회전 | X |
| 12 | 1 | 1 | 0 | 1 | 0 | 직진 | 좌회전 | X |
| 13 | 0 | 0 | 1 | 1 | 0 | 좌회전 | X | X |
| 14 | 1 | 0 | 1 | 1 | 0 | 좌회전 | X | X |
| 15 | 0 | 1 | 1 | 1 | 0 | 직진 | 좌회전 | X |
| 16 | 1 | 1 | 1 | 1 | 0 | 직진 | 좌회전 | X |
| 17 | 0 | 0 | 0 | 0 | 1 | 우회전 | X | X |
| 18 | 1 | 0 | 0 | 0 | 1 | 우회전 | X | X |
| 19 | 0 | 1 | 0 | 0 | 1 | 직진 | 우회전 | X |
| 20 | 1 | 1 | 0 | 0 | 1 | 직진 | 좌회전 | 우회전 |
| 21 | 0 | 0 | 1 | 0 | 1 | 우회전 | X | X |
| 22 | 1 | 0 | 1 | 0 | 1 | 우회전 | X | X |
| 23 | 0 | 1 | 1 | 0 | 1 | 직진 | 우회전 | X |
| 24 | 1 | 1 | 1 | 0 | 1 | 직진 | 좌회전 | 우회전 |
| 25 | 0 | 0 | 0 | 1 | 1 | 좌회전 | 우회전 | X |
| 26 | 1 | 0 | 0 | 1 | 1 | 좌회전 | 우회전 | X |
| 27 | 0 | 1 | 0 | 1 | 1 | 직진 | 좌회전 | 우회전 |
| 28 | 1 | 1 | 0 | 1 | 1 | 직진 | 좌회전 | 우회전 |
| 29 | 0 | 0 | 1 | 1 | 1 | 좌회전 | 우회전 | X |
| 30 | 1 | 0 | 1 | 1 | 1 | 좌회전 | 우회전 | X |
| 31 | 0 | 1 | 1 | 1 | 1 | 직진 | 우회전 | X |
| 32 | 1 | 1 | 1 | 1 | 1 | 직진 | 좌회전 | 우회전 |

표 3. 먹이상황 별 인공개체의 최적 행동 분석

1. 1단위시간 안에 먹이를 먹을 수 있을 경우: 먹이쪽으로 이동
2. 1단위시간 안에 먹이를 먹을 수 없을 경우: 먹이가 있는 방향으로 이동
3. 아무런 먹이도 없는 경우: 직진

1번의 경우 인공개체는 1단위시간 안에 먹이를 먹을 수 있기 때문에 먹이가 있는 방향으로 이동하는 것이 최적의 행동이다. 2번의 경우 T시간단위에 있는 인공개체는 T+1단위시간 안에 먹이를 먹을 수 없기 때문에 T+2단위시간 내에 먹이를 먹을 수 있는 방향으로 이동하는 것이 최적의 행동이다. 3번의 경우 주변에 아무런 먹이가 없기 때문에 먹이를 찾도록 움직여야하기 때문에 직진하는 것이 최적의 행동이다. 위의 세 먹이 상황에 따라 얻은 인공개체의 최적의 행동은 표 3에서 확인할 수 있다.

표 3에서 얻은 행동을 기준으로 시뮬레이션 상의 모든 인공개체의 적합도 평가를 하였다. 적합도 평가는 한 세대가 끝날 때마다 살아있는 인공개체의 내부로직의 32개 입력에 대한 행동이 표 3의 case중 적어도 하나에 속하면 +1점을, 하나도 속하지 아니하면 +0점을 주어 평가하였다. 인공개체가 가질 수 있는 최고 적합도 점수는 32점이여 만약 32점 내부로직을 갖는 인공개체를 찾았을 경우 시뮬레이션을 종료하였다.

실험에 사용된 파라미터는 표 4과 같다. CC의 셀 공간은 35x35로 설정하였다. 인공개체의 초기 개수는 셀 공간의 한쪽 크기 35를 2로 나눈 후 반 내림한 17을 사용하였다. 특별한 이유가 있는 것이 아니고 인공개체의 움직임을 적절히 관찰하기 위하여 셀 공간 크기와 비례하여 설정한 것이다. 인공개체의 최소 개수는 초기 인공개체수와 동일하도록 설정하였고, 인공개체의 최대 개수는 인공개체의 초기 개수의 2.5배에 반올림한 값을 사용하였다. 먹이의 개수는 인공개체의 초기 개수의 10배에 해당하는 값으로 설정하였다. 인공개체진화에서 적당한 먹이의 분포는 중요하다. 먹이가 부족한 경우 굶어죽는 인공개체가 너무 많이 발생하며 먹이가 너무 많은 경우 경쟁력이 없는 인공개체도 살아남아 진화에 도움이 되지 않는다. 그러나 인공개체 개수에 비하여 어느 정도의 먹이가 있는 것이 적당한지는 알 수 없다. 그래서 본 논문에서는 경험적인 값으로 인공개체 초기 개수에 10배를 사용하였다. 최대 세대는 200으로 다른 값들과 연관성이 없는 고정 값이다.

표 4. 기억회로가 없는 인공개체 실험 파라미터

| | |
|---|---|
| CC의 공간 크기 설정 | 35 x 35 |
| 개체 / 먹이의 초기 개수 설정 | 17 / 170 |
| 개체 / 먹이의 최대 개수 설정 | 43 / 170 |
| 개체 / 먹이의 최소 개수 설정 | 17 / 170 |
| 시간단위 : 세대 비율 설정 | 35 : 1 |
| 최대 세대 설정<br>(시뮬레이션 완료 조건) | 200 |
| 돌연변이 확률 (교차점 당) | 0.05 |

최고 적합도 32를 갖는 인공개체를 찾지 못한 경우 200세대에서 시뮬레이션이 종료한다. 그러나 실험결과 대부분의 경우 200세대 이전에 최고 적합도를 갖는 인공개체가 생성되었다. 이번 실험에서 사용된 단위시간과 세대의 비율은 35:1이다. 그렇기 때문에 35단위시간에서 단위시간은 0으로 초기화되며 세대는 +1이 된다. 자식 개체를 생성할 때 돌연변이 확률은 행위결정 로직프레임의 교차점 하나당 0.05의 확률을 사용하였다.

시뮬레이션이 시작되면 CC의 셀 공간 내에 인공개체와 먹이가 임의의 위치에 생성된다. 다만, 먹이와 인공개체가 동일한 셀에 생성되지는 않는다. 각 인공개체의 초기 내부로직은 무작위로 생성되므로 초기에는 대부분의 인공개체가 다른 로직을 갖는다. 초기에 인공개체 로직은 임의로 만들어져서 대부분 적합한 행동을 하지 못한다. 그 경우 먹이를 먹지 못해 죽게 된다. 인공개체가 죽는 조건은 다음과 같았다.

1. 먹이를 하나도 먹지 못 한 인공개체
2. 5세대 이상을 살은 인공 개체
3. 적합도가 낮은 개체

인공개체가 한 세대를 살며 적어도 하나 이상의 먹이를 먹지 못할 경우 1번 조건에 의해 굶어 죽게 된다. 또한 5세대 이상을 살아온 인공개체는 2번 조건에 의해 늙어 죽게 된다. 3번 조건의 경우 인공개체가 포화 상태에 이렀을 때 적용되는 조건이다. 어느 정도의 세대가 지나면 인공개체들은 대부분 한 개 이상의 먹

이를잘 찾아 먹게 된다. 이 경우 대다수의 인공개체가 번식을 하기 때문에 인공개체수가 증가하게 되는데 파라미터의 최대 개체 수를 초과하게 되면 3번 조건이 적용되어 인공개체를 죽이게 된다. 이 경우 인공개체의 적합도를 높은 순으로 정렬하여 낮은 적합도의 인공개체들 중 최대 개체 수를 초과한 만큼의 인공개체를

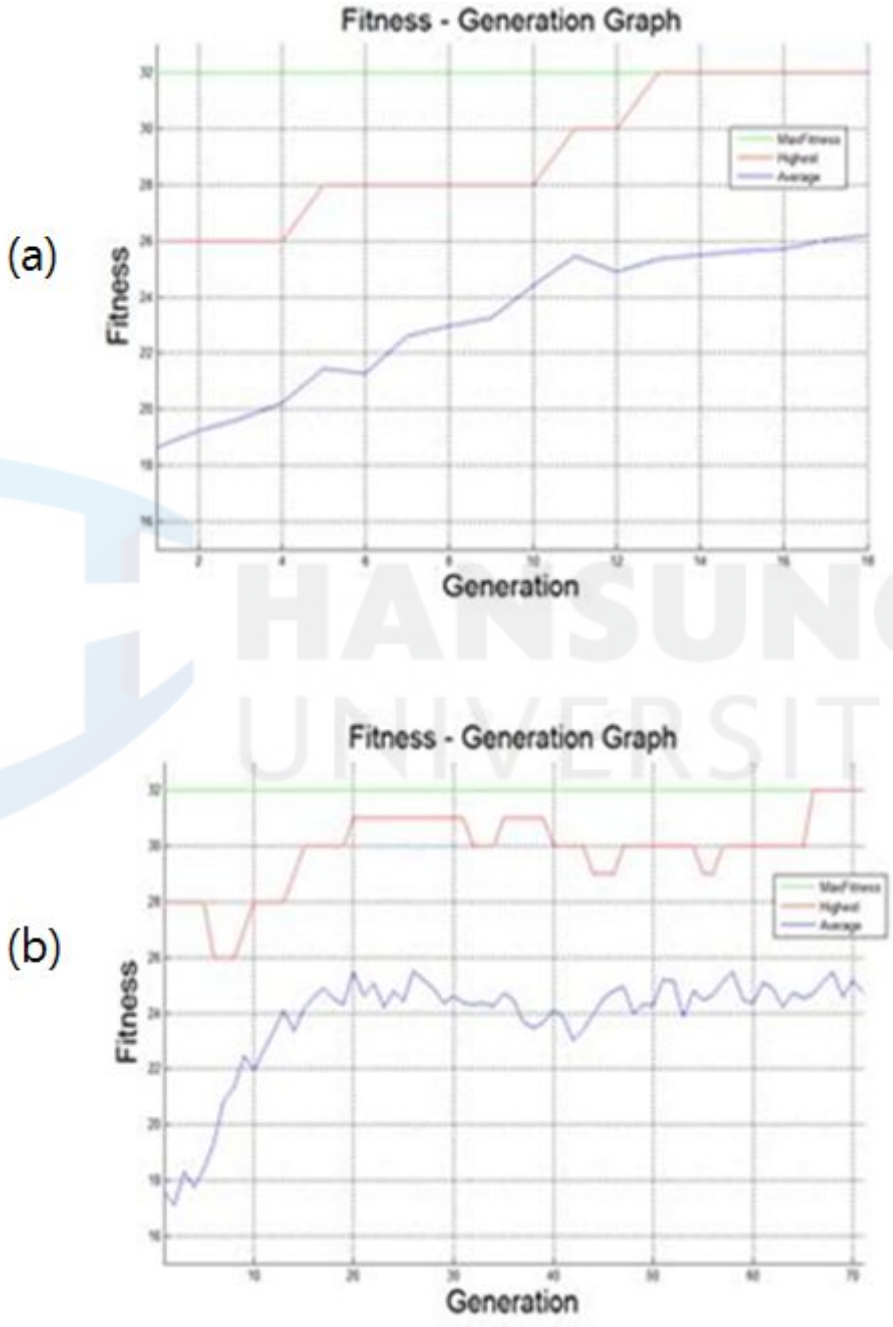



그림 13. 적합도 평가를 통한 진화 (a) 안정적인 진화 (b) 불안정적인 진화

죽이게 된다. 그리고 모든 인공개체는 세대가 끝나고 다음 세대로 넘어갈 때 이며 이는 단위시간이 35에서 0으로 초기화 되는 시기와 동일하다. 인공개체가 사는 조건은 다음과 같다.

1. 한 세대 내에서 먹이를 하나 이상 먹은 경우
2. 적합도가 높은 경우
3. 부모 개체에 의해 생성된 자식 개체의 경우

인공개체의 번식은 매 세대가 끝나고 시작할 때 살아있는 개체를 대상으로 이뤄진다. 세대가 진행되면서 먹이를 잘 먹는 인공개체들은 살아남아 자식을 생성하므로 점점 더 먹이를 잘 먹는 인공개체들로 진화된다. 최적의 인공개체로 진화된 경우 시뮬레이션의 목적을 달성한 것이기 때문에 시뮬레이션을 종료하면 된다. 만약 최적의 진화를 모르는 경우에는 이 방법으로 종료를 하기 어렵기 때문에 보통 최대 세대수로 시뮬레이션을 종료한다. 본 연구에서는 최대 적합도 32점을 알기 때문에 최대 적합도를 갖는 개체가 발생한 경우 시뮬레이션을 종료한다.

인공개체가 잘 진화되는지 여러 번의 실험을 통하여 확인하였다. 확인 결과 세대가 지나가면서 점점 더 높은 적합도를 갖는 인공개체로 진화되는 것을 볼 수 있었다. 초기에 생성된 인공개체의 내부로직들은 대략 평균 26점의 적합도를 가졌고, 이 인공개체들이 32점의 적합도를 갖는데 걸린 평균 세대수는 대략 36세대이다. 그림 13(a)의 그래프는 하나의 시뮬레이션에서의 최대적합도와 평균 적합도를 보여준다. 그림에서 보듯이 평균적합도가 세대가 지날수록 안정적으로 상승하고 있고 13세대에서 최대 적합도 32를 갖는 인공개체로 진화하였다. 그림 13(b)의 그래프에서는 상대적으로 진화가 더딘 결과를 보여준다. 또한 최대 적합도를 유지하지 못하는 것을 보여준다. 이것은 이전 세대에서 최대 적합도를 갖는 인공개체가 다음 세대에서 자식을 생성하고 죽은 경우, 자식에 발생한 변이가 인공개체의 적합도를 낮추어 최대 적합도를 유지하지 못하는 경우이다. 변이는 더 좋은 인공개체를 찾기 위해 필요하지만 반드시 좋은 방향으로의 변이만 발생하는 것은 아니다. 특히 진화가 많이 일어난 경우 대부분의 변이는 좋지 않은 쪽으로 일어난다.

그림 14은 최대 적합도를 갖는 인공개체의 행동을 살펴보기 위해 최대 적합도를

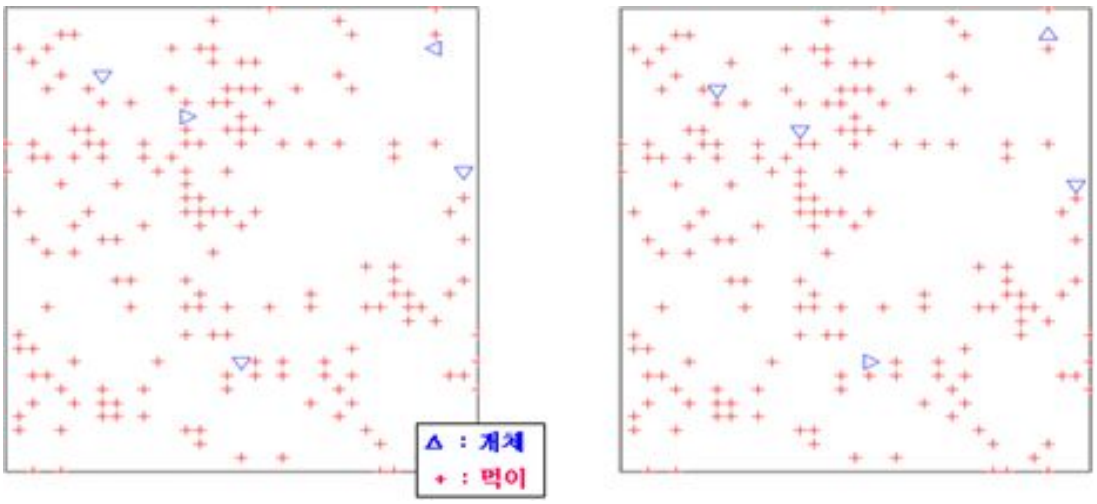



그림 14. 진화한 인공개체의 행동

갖는 인공개체가 발생한 후에 해당 인공개체의 내부로직을 5개의 인공개체로 테스트해본 결과화면이다. 인공개체의 삼각 모형 중 동, 서, 남, 북을 향하고 있는 방향이 현재 인공개체의 이동 방향이다(자세한 그림은 그림 15 참조). 그림 14의 왼쪽은 시간 t에서, 오른쪽은 시간 t+1에서의 실행화면이다. 3개의 인공개체가 먹이를 먹기 위해 방향을 튼 모습을 볼 수 있다. 맨 오른쪽 위의 인공개체의 경우 지난 간 자리에 먹이가 새로 생성된 것을 보여준다. 수십 세대 동안 살펴본 결과 먹이를 잘 찾아서 먹는 것을 살펴볼 수 있었다. 실험에서 최대 적합도를 갖는 두 인공개체의 내부로직을 살펴보니 다음과 같았다.

관찰 인공개체 1

1. O1 = N5' + N2N3N4'
2. O2 = N4' + N5

관찰 인공개체 2

3. O1 = N5'
4. O2 = N1'N3'N4' + N2 + N5

최대 적합도는 행동으로 결정하는 것이기에 최대 적합도를 보이는 입/출력 진리표가 다수 개 존재한다. 왜냐하면 동일한 상황에서 다수 개의 적합한 행동이 있기 때문이다. 예를 들면 N1~N5에 모두 먹이가 있는 경우 N1~N5 어느 곳으로 가던지 적합한 행동이 된다. 또한 하나의 진리표를 구현하는 것도 다수의 로직이 가능하다. 그러므로 최대 적합도를 보이는 인공개체의 내부로직은 매우 많

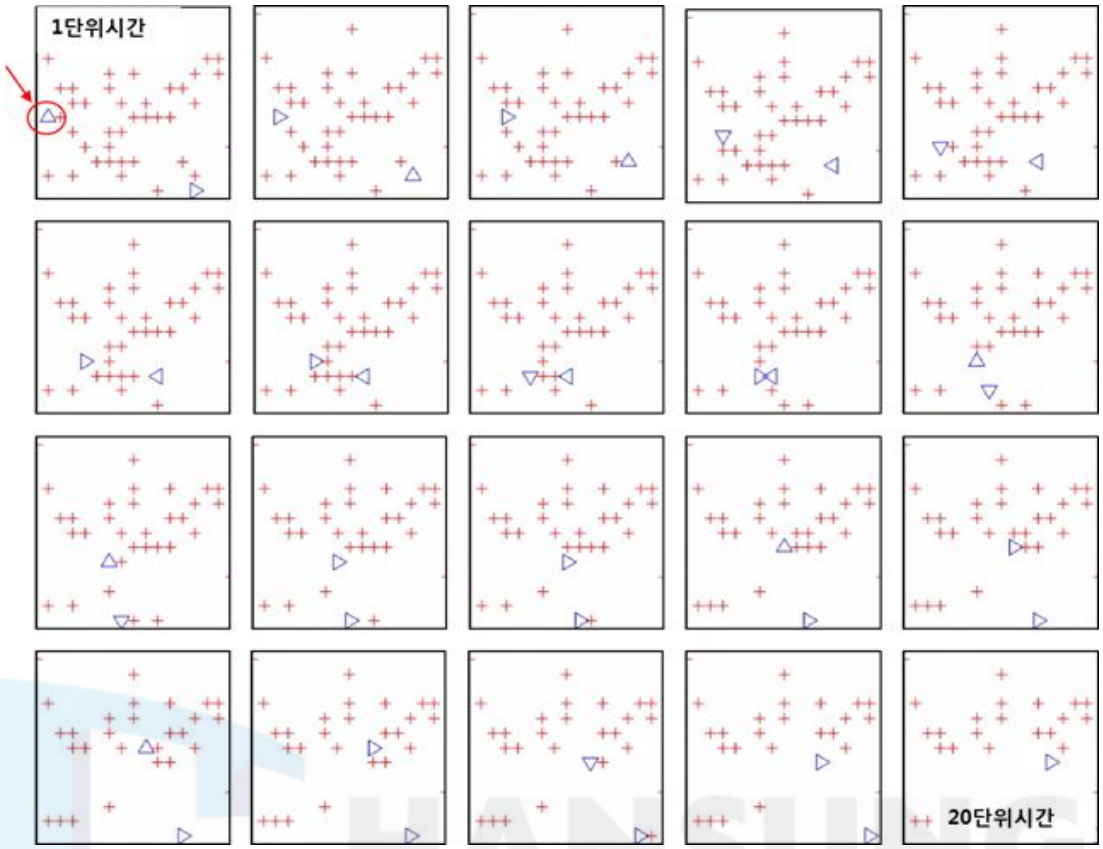



그림 15. 적합도 평가를 통한 인공개체의 진화 후 행동

이 존재할 수 있다.

표 5는 관찰 개체 1, 2의 행동 결정표를 나타낸다. N1~N5는 입력이고 O1과 O2는 입력에 대한 두 OR게이트의 출력이다. 출력을 통하여 그림 14처럼 각 인공개체의 행동이 결정된다. 두 인공개체는 몇 개의 경우에 서로 다른 행동을 결정하는 것을 보여준다. 이는 두 인공개체가 서로 다르게 진화해 왔지만 둘 다 환경에 적합한 형태로 진화했음을 보여준다. 이런 결과로 보았을 때 본 논문에서 제안한 셀 우주가 셀 공간상에서 진화적 모델링 및 시뮬레이션 프레임워크로 유용함을 볼 수 있었다.

| 입력 | | | | | Sample 1 | | | Sample 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| N1 | N2 | N3 | N4 | N5 | O1 | O2 | Action | O1 | O2 | Action |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 직진 | 1 | 0 | 좌회전 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 직진 | 1 | 0 | 좌회전 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 직진 | 1 | 1 | 직진 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 0 | 좌회전 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 0 | 좌회전 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 1 | 직진 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 1 | 직진 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 0 | 좌회전 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 0 | 좌회전 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 1 | 직진 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 좌회전 | 1 | 1 | 직진 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 직진 | 0 | 1 | 우회전 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 직진 | 0 | 1 | 우회전 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 우회전 | 0 | 1 | 우회전 |

표 5. 관찰 개체의 행동 결정 표

### 4.1.2 다양한 먹이구조에서 기억회로가 없는 인공개체의 진화

우리는 4.1.1절을 통하여 셀 우주에서 제안하는 진화 시뮬레이션 프레임워크가 잘 동작 하는지 확인하였다. 4.1.1절의 먹이들은 임의의 위치에서 재생성 되었기 때문에 특별한 구조를 가지고 있지 않았다. 우리는 실험을 조금 더 심화하기 위하여 먹이를 구조화 시켜 고정된 위치에서 재생성 되도록 설정한 후 실험을 하였다. 또한 이전 절에서는 개체수가 포화상태에 이르면 적합도 평가를 통해 인공개체를 죽였지만 이번 절에서는 적합도 평가 대신 단순히 먹이를 먹은 개수를 따져 실험하였다. 이 경우 먹이를 많이 먹은 인공개체들이 살게 되며 비교적 덜 먹은 인공개체들은 살지 못하게 설정하였다.

우리는 먹이가 그림 16와 같은 특정한 구조를 가지고 생성될 때 인공개체의 행동을 볼 수 있는 실험을 하였다. 그림은 더욱 다양한 먹이구조에서 실험하였으나 그 중 대표적인 2개의 먹이구조만 보여주고 있다. 그림 16(a)는 9개의 먹이가 3x3정사각형구조를 갖고 있으며 그림 16(b)는 직렬로 4개의 먹이가 연결된 구조를 가지고 있다.

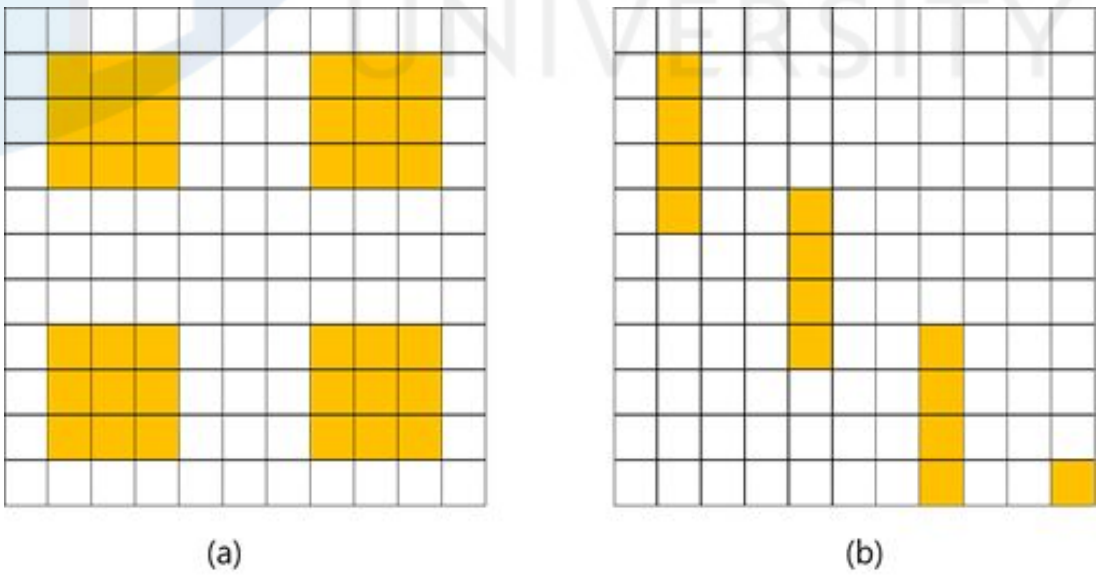



그림 16. 다양한 먹이구조

3x3 정사각형 구조인 그림 16(a)에서 인공개체를 진화시켜 본 결과 대부분의

인공개체는 그림 16(a)와 같거나 비슷한 행동을 하였다. 3x3의 정사각형 먹이 중 1개를 제외하고 나머지 먹이를 외곽으로부터 내부로 한바퀴 회전하며 먹는 형태로 진화하였다. 이 결과를 통하여 32점의 행동을 기준으로 진화에 있어서 적합도 평가를 하지 않더라도 인공개체가 먹이에 대해 잘 반응하는 쪽으로 진화함을 확인하였다. 그림 17은 이 행동을 보여주고 있다. 그림의 붉은색 점에서 시작한 인공개체는 정사각형의 가장 외곽의 먹이를 한바퀴 돌면서 8개중 7개를 먹은 후 가운데의 먹이를 먹고 직진하여 그 다음의 정사각형을 찾는 과정을 보여주고 있다.

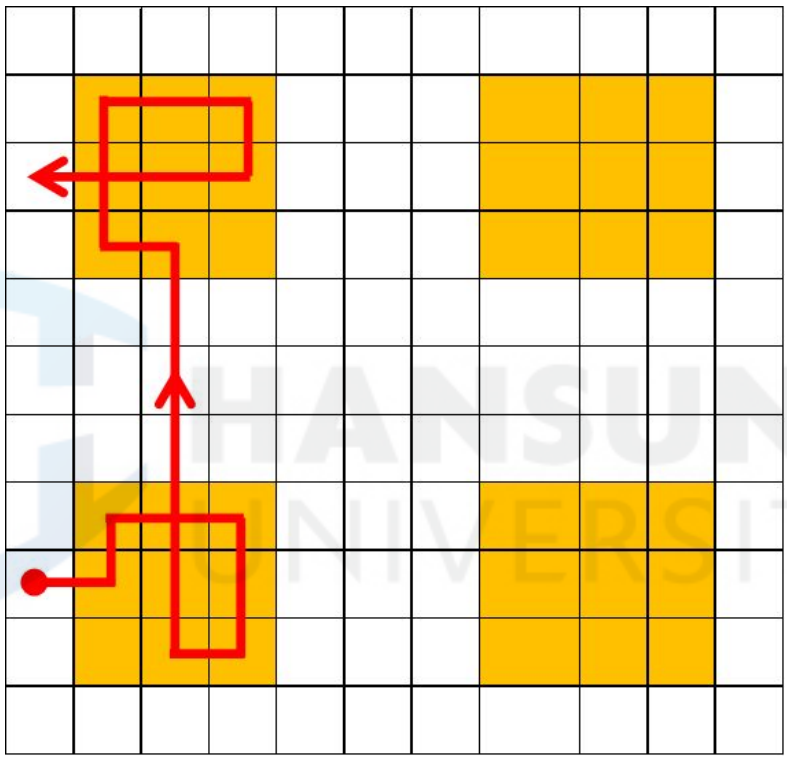


그림 17. 3x3 정사각형 먹이구조에서의 행동

우리는 기억회로가 없이 먹이 상황에 대해서 행동하는 인공개체의 한계를 실험하기 위해 그림 16(b)의 먹이구조를 설계하였다. 4개의 직렬 먹이를 인공개체가 수직 방향으로 먹을 때 먹이의 끝나는 점에서 인공개체의 이웃 셀은 모두 비어있게 된다. 이 경우 인공개체가 할 수 있는 4개의 행동 중 하나를 하게 되는데 어떠한 행동을 하더라도 인공개체의 이웃 셀은 비어있게 된다. 예를 들어 T시간단계의 먹이의 끝점에서 직진을 한 인공개체는 T+1시간단계에서도 동일한 먹이상황에 놓이게 되어 이전시간단계인 T때의 행동을 하게 된다. 마찬가지로 T시간단

계의 먹이의 끝점에서 우회전을 한 인공개체는 T+1때 동일한 먹이상황에 놓이게 되어 다시 한번 우회전을 하게 된다. 두 번 우회전을 한 인공개체는 이전에 먹은 동일한 먹이구조를 먹을 것 같지만 이미 해당 먹이구조는 인공개체에 의해 먹혀왔기 때문에 U턴을 하더라도 아무런 먹이가 없는 상황에 놓이게 된다. 그렇기 때문에 기억회로가 없는 인공개체가 그림 16(b)의 먹이구조에서 먹이가 재생성 되지 않는 한 최대로 먹을 수 있는 먹이 수는 4개이다. 그림 18은 진화한 인공개체의 행동이다. 위에서 설명한 것처럼 인공개체가 4개의 직렬 먹이를 먹은 후 직진을 하여 옆에 있는 다른 먹이를 먹지 못하는 것을 볼 수 있다.

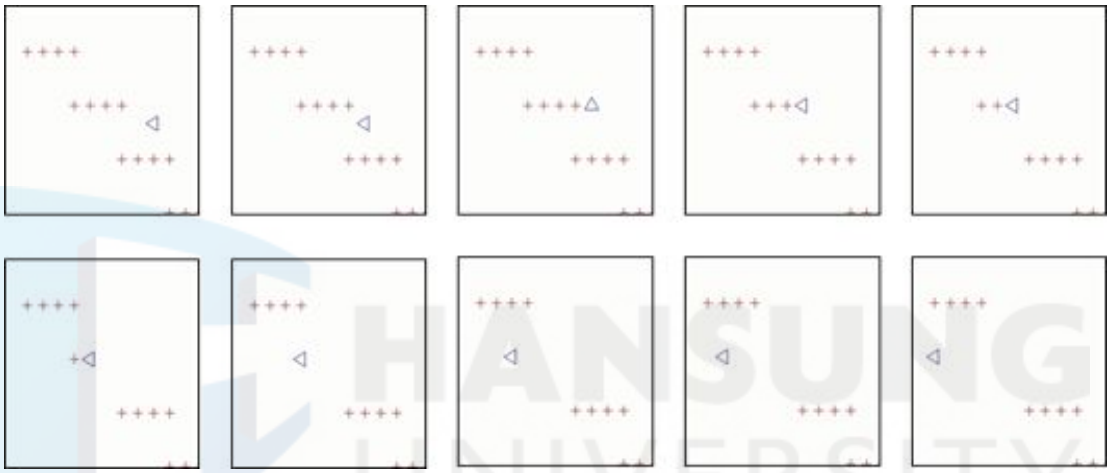


그림 18. 기억회로가 없는 인공개체의 한계

우리는 셀 우주의 인공개체-먹이 모델의 검증을 위하여 기억회로가 없는 인공개체를 적합도 평가로 진화시킨 후 먹이를 잘 먹도록 진화하는지 확인하였다. 또한 행위결정 로직프레임의 내부로직이 모든 입력인 32개에 대하여 가장 최적의 출력을 내보낼 수 있는지 확인하였다. 그 결과 인공개체는 5개의 이웃 셀에 대하여 먹이가 존재할 시 가장 최적화된 방향으로 이동하며 먹이를 먹는 것을 확인하였고 이 인공개체가 32개의 먹이 상황에 대해 모두 먹이를 잘 먹는 쪽으로 이동하는 내부로직을 가지고 있음을 확인하였다.

또한 프레임워크 상에서 먹이들이 특정한 구조를 가지고 태어날 경우 인공개체가 그 패턴에 대해 최적화된 방향으로 이동하는지 확인하여 보았다. 더하여서 먹이구조를 더 복잡한 형태로 구성하였을 때에도 인공개체가 먹이를 잘 먹을 수 있는지 확인하였다. 그 결과 비교적 복잡하지 않은 먹이구조는 인공개체가 인식할

수 있었으나 조금 복잡한 구조에서는 한계가 있음을 확인하였다. 우리는 이 한계를 개선하기 위하여 행위결정 로직프레임에 인공개체가 과거의 행동을 입력으로 받아 행동할 수 있도록 설계하여 실험을 해 보았고 그 과정과 결과를 4.2 절에서 기술하였다. 또한 특정한 먹이구조를 기준으로 인공개체가 기억할 행동의 수가 다를 때 몇 단계의 행동을 기억하는 것이 가장 최적인지 실험을 통해 알아보았다.

## 4.2 기억회로가 있는 경우

셀 우주의 인공개체-먹이 모델에서 기억회로는 행위결정 로직프레임의 한 부분이다. 이 기억회로를 통해 인공개체는 과거에 자신이 했던 행동을 현재의 입력으로 되먹임 받아 현재의 행동을 결정할 수 있다. 그렇기 때문에 인공개체가 똑같은 먹이구조를 가질 수밖에 없는 상황에 놓여있더라도 과거에 한 행동에 따라 현재의 행동이 바뀌기 때문에 먹이를 찾아 먹을 수 있게 된다. 이 절에서는 기억회로를 통하여 4.1절의 인공개체의 한계점을 보완할 수 있는지 실험을 통해 알아보았다.

우리는 기억회로가 있는 인공개체와 없는 인공개체 행동의 차이를 알아보기 위하여 4.1.2절에서 실험한 먹이 구조에서 동일하게 진화시킨 후 인공개체가 어떻게 행동하는지, 그리고 4.1절의 인공개체와 행동이 어떻게 다른지 차이를 분석하였다. 또한 특정한 패턴에서 인공개체가 자신의 행동을 몇 단계 기억하는 것이 가장 최적인지를 실험하였고 그 결과를 분석하였다.

### 4.2.1 기억회로가 있는 인공개체의 행동 분석

기억회로가 있는 인공개체와 없는 인공개체의 행동의 차이를 알아보기 위하여 4.1절에서 사용한 2개의 먹이구조에서 동일하게 진화시켜 보았다. 이번 실험에서 사용한 인공개체의 행위결정 로직프레임은 그림 11과 같이 4시간단계까지 기억하는 프레임을 사용하였다.

위에서 설명한 바와 같이 기억회로를 갖는 인공개체는 이웃 셀의 먹이상황을

입력받음과 동시에 자신의 과거 행동을 되먹임 받아 현재의 행동을 결정할 수 있다. 그렇기 때문에 더욱 다양한 행동을 할 수 있게 되어 복잡한 먹이구조에서도 먹이를 잘 먹는 행동을 보일 수 있게 된다. 이를 확인하기 위해 우리는 기억회로가 없는 인공개체의 행동과 비교해 보는 실험을 하였다. 실험을 위해 4.1.2절의 그림 16(a)와 (b)인 복잡한 먹이구조에서 동일한 조건으로 기억회로를 갖는 인공개체를 진화시켜 보았다. 그림 16(a)에서의 진화는 기억이 없는 회로와 거의 비슷한 형태로 진화하였다. 기억회로를 갖는 인공개체는 9개의 먹이를 모두 다 먹고 다음 9개의 먹이를 찾아가는 행동을 보였다. 하지만 이 행동을 분석해본 결과 기억회로가 없더라도 동일한 행동을 보일 수 있다는 결과를 얻었다. 그렇기 때문에 그림 16(a)의 구조에서 진화시킨 기억회로가 있는 인공개체와 없는 인공개체의 행동의 차이는 없다고 판단하였다.

하지만 그림 16(b)에서 진화를 시킨 기억회로가 있는 인공개체의 행동은 많이 달랐다. 그림 19은 그림 16(b)에서 진화하였을 때의 기억회로가 있는 인공개체의 행동을 보여준다. 4개의 직렬로 되어있는 먹이의 시작점에 도착한 인공개체는 그림 19의 2번 사진처럼 좌회전을 하며 먹이를 4개를 먹게 된다. 그리고 끝점에 도달하였을 때 인공개체는 4번 사진처럼 아무런 먹이가 없는 상황에 놓인다. 이 때 5번 사진에서 우회전을 하였지만 아무런 먹이를 찾지 못 하였다. 만약 이 개체가 기억회로가 없는 인공개체라면 6번 그림에서 하는 행동이 또다시 우회전이 될 것이다. 하지만 기억회로를 갖는 인공개체는 6번 사진처럼 직진을 하여 다음 4개의 직렬 먹이구조를 찾는 모습을 볼 수 있었다. 6번 사진에서 먹이를 찾은 인공개체는 다시 4개의 먹이를 먹으며 이동해 나아간다.

기억회로가 있는 인공개체가 이와 같은 행동이 어떻게 가능한지 분석하기 위하여 인공개체의 입력의 변화를 그림 21을 통해 확인해 보았다. 표 6는 그림의 1~10번의 대한공개체의 한 행동을 보여주고 있다 이 그림에서 주요 시간은 4번과 5번이다. 4번과5의 먹이 입력은 모두 0으로 동일하다. 그렇기 때문에 기억회로가 없는 인공개체는 5번과 6번에서 동일한 행동을 보였다. 하지만 기억회로가 있는 인공개체의 경우 5번에서는 우회전 하였으나 6번에서는 직진하는 모습을 보았다.

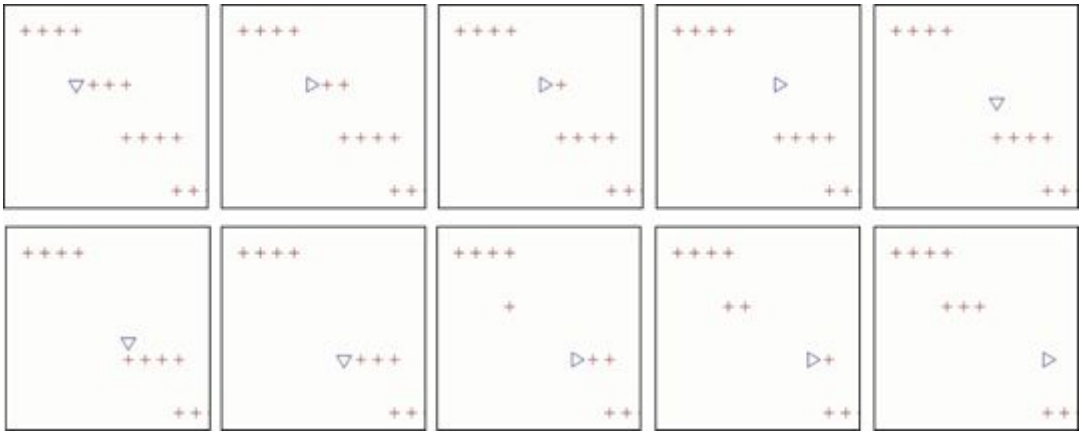


그림 19. 기억회로가 있는 인공개체의 진화

### 4.2.2 기억 단계가 서로 다른 인공개체의 비교

우리는 4.2.1 절의 실험을 통하여 기억회로가 있는 인공개체는 복잡한 먹이환경에서도 다양한 행동의 조합으로 먹이를 잘 찾아먹음을 확인하였다. 하지만 기억회로가 있다고 무조건 좋다고 할 수는 없다. 예를 들어 그림 16(a)의 먹이 환경에서 진화한 두 인공개체의 행동을 비교하여 보았을 때 결과 적으로는 기억회로가 있는 인공개체가 더 좋았지만 거의 비슷한 행동을 보였기 때문이다. 그리고 분석 결과  두 인공개체의 행동이 기억회로가 있거나 없더라도 모두 발생할 수

| 시간단계 | N1 | N2 | N3 | N4 | N5 | t-4 | t-3 | t-2 | t-1 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 직진 | 우회전 | 직진 | 직진 |
| 2 | 0 | 1 | 0 | 0 | 0 | 우회전 | 직진 | 직진 | 우회전 |
| 3 | 0 | 1 | 0 | 0 | 0 | 직진 | 직진 | 우회전 | 직진 |
| 4 | 0 | 0 | 0 | 0 | 0 | 직진 | 우회전 | 직진 | 직진 |
| 5 | 0 | 0 | 0 | 0 | 0 | 우회전 | 직진 | 직진 | 우회전 |
| 6 | 0 | 1 | 0 | 0 | 0 | 직진 | 직진 | 우회전 | 직진 |
| 7 | 0 | 0 | 0 | 1 | 0 | 직진 | 우회전 | 직진 | 직진 |
| 8 | 0 | 1 | 0 | 0 | 0 | 우회전 | 직진 | 직진 | 우회전 |
| 9 | 0 | 1 | 0 | 0 | 0 | 직진 | 직진 | 우회전 | 직진 |
| 10 | 0 | 0 | 0 | 0 | 0 | 직진 | 우회전 | 직진 | 직진 |

표 6. 기억회로를 갖는 인공개체의 입력의 변화

있는 행동이었기 때문이다. 이러한 경우 오히려 복잡한 기억회로가 있는 인공개체보다 없는 인공개체가 더 유리할 수 있다.

행위결정 로직프레임의 입력은 비트단위이기 때문에 2개의 경우의 수를 가지고 있으며 1개의 기억단계를 갖는 기억회로를 구성하기 위해서 2비트의 입력이 추가되어 $2^2$=4개의 경우의 수가 추가된다. 이는 행위결정 로직프레임의 먹이입력의 경우의 수인 $2^5$에 1개의 기억단계의 경우의 수인 $2^2$가 곱해지는 경우이므로 입력의 경우의 수는 $2^7$인 128개가 된다. 마찬가지로 2개의 기억단계를 갖는 기억회로를 구성하기 위해서는 4비트의 입력이 추가되므로 입력의 경우의 수는 $2^9$인 512개가 된다. 이처럼 기억단계가 하나 늘어날 때 마다 입력의 경우의 수는 4배로 증가하게 되어 너무 많은 기억단계를 갖는 인공개체를 사용할 경우 입력의 경우의 수가 매우 늘어나게 되고 사용하는데 어려움이 있을 수 있다.

기억회로의 기억단계가 늘어날수록 내부로직의 탐색 공간 또한 늘어나게 된다. 내부로직의 탐색공간은 행위결정 로직프레임의 입력과 AND게이트들의 교차점이다. 그리고 행위결정 로직프레임은 1비트의 입력에 대해 16개의 교차점을 갖는 구조를 가지고 있다. 기억회로가 없는 행위결정 로직프레임의 경우 5비트의 먹이입력을 갖기 때문에 탐색공간의 크기는 5x16=80개가 된다. 또한 하나의 탐색공간은 접점이 있거나 없는 경우를 갖기 때문에 2개의 경우를 가질 수 있다. 즉 기억회로가 없는 행위결정 로직프레임의 탐색공간이 가질 수 있는 모든 경우의 수는 $(5*16)^2$개이며 이는 매우 큰 수치이다. 이미 충분히 큰 탐색공간을 갖는 행위결정 로직프레임에 2비트의 입력을 갖는 한 단계의 기억회로가 추가된다면 탐색공간의 경우의 수는 $((2+5)*16)^2$개로 늘어나게 된다. 이처럼 기억회로의 기억단계가 하나 늘어날 때 마다 탐색공간에 미치는 영향이 크다.

입력 경우의 증가와 탐색 공간의 증가는 결국 진화하는데 걸리는 시간에 영향을 준다. 그렇기 때문에 그림 16(a)와 같이 단순한 먹이구조를 갖는 환경에서는 오히려 기억회로를 추가하는 것이 성능에 영향을 미칠 수 있다. 또한 기억회로가 있다 하더라도 기억단계를 얼마나 설정 할 것인가에 따라 성능이 달라진다. 우리는 기억회로의 기억단계별 성능의 차이를 알아보기 위하여 특정한 먹이환경을 구성하여 기억회로의 기억단계가 서로 다를 때 어떤 인공개체가 가장 최적인지를

표 7. 기억회로를 갖는 인공개체 실험 파라미터

| 마름모 먹이구조 파라미터 | |
|---|---|
| 최대세대 | 한 개체의 멸종 |
| 공간 크기 | 228x228 |
| 최소 개체수 | 100 |
| 최대 개체수 | 300 |
| 먹이로직 초기 접점 결정 확률 | 3 / 80 |
| 기억로직 초기 접점 결정 확률 | 3 / 128 |
| 변이확률 | 1 / 총교차점 수 |
| 먹이재생성 시간 | 10 시간단위 |
| 시간단위 : 세대 배율 | 35 : 1 |

두 실험을 통해 알아보았다.

실험에 사용된 파라미터 설정은 표 7과 같다. 인공개체가 살아가는 공간은 228x228로 설정하였다. 최소 인공개체수는 초기에 생성하는 모든 타입의 인공개체 수를 말한다. 최대 개체수는 인공개체의 종류에 상관없이 한 실험에서 최대로 존재할 수 있는 인공개체수를 말한다. 실험초기에는 각 인공개체 타입 별로 최소 개체수가 생성되어 진화하면서 자식을 낳기 때문에 인공개체수가 증가한다. 그러나 전체 인공개체수가 최대 인공개체수를 넘으면 각 인공개체가 먹은 먹이수가 높은 인공개체 순으로 정렬하여 최대 인공개체수 이하의 인공개체는 죽여서 최대 인공개체수를 유지한다. 인공개체는 세대가 지남에 따라서 각 인공개체의 행위결정 로직프레임의 내부로직이 진화하게 된다. 초기 행위결정 로직프레임의 접점은 무작위로 결정되는데 접점이 생성될 확률은 3/80으로 주었다. 행동을 기억하는 부분의 초기 접점 확률은 3/128로 주었다. 자식을 생성할 때는 부모의 내부로직에서 변이를 주어 상속하는데 그 변이 확률은 (1/총교차점 수)로 주었다. 인공개체가 먹이를 먹으면 점점 더 먹이가 사라져서 대부분의 개체가 굶어죽게 된다. 이를 방지하기 위하여 먹이는 주기적으로 재생되도록 만들었는데 그 주기는 10개의 단위시간이다. 인공개체가 자식을 생성하거나 굶어죽는 것은 세대별로 이루어지는데 35개의 단위시간이 지난 경우 한 세대가 지난 것으로 한다. 그리고 개체가 어떤 단계까지 기억할 수 있느냐에 따라서 각 개체 타입에 명칭을 부여하였다. T0 는 기억이 없는 개체타입을 가리키며 T4 는 4개의 이전 행동까지 기억하

는 개체 타입이다.

그림 20. 마름모 먹이구조

실험에 사용된 먹이구조는 마름모형 구조이다. 마름모형 먹이구조는 그림 20과 같다. 전체적인 구조를 봤을 때 마름모 형태를 보이지만 마름모의 선분이 되는 위치에서 먹이가 대각선으로 한 칸씩 떨어져서 존재하는 것을 확인할 수 있다. 한 칸씩 떨어져 있기 때문에 어느 정도 행동을 기억하는 인공 개체가 좋게 진화할 것으로 예상하여 마름모형 먹이구조를 선택하였다. 특히 마름모 먹이구조에 최적화된 기억단계를 갖는 인공개체는 과거행동과 현재의 상황을 통하여 어떤 행동을 해야 먹이가 있는 곳으로 갈 수 있는지를 판단할 수 있을 것이다.

인공개체가 몇 단계까지 자신의 과거행동을 기억해야 최적인지는 먹이패턴에 따라 다르다. 일반적으로 복잡한 먹이패턴을 위해서는 많은 단계의 과거행동까지 기억하는 것이 좋다. 다만, 과거행동을 기억하기 위한 새로운 입력이 주어지기 때문에 기억회로의 최적화 입장에서는 탐색공간이 늘어나는 단점이 발생한다. 그러므로 과거행동을 기억하는 단계가 많을수록 무조건 좋은 것이 아니며, 먹이구조

의 복잡도와 행위결정로직의 탐색공간 크기에 따라서 가장 적절한 단계가 결정된다. 우리는 이러한 것을 실험으로 입증하기 위하여 4단계까지 과거 입력을 받는 인공개체를 이용하여 실험하였다.

위에서 설명한 파라미터와 먹이구조로 우리는 두 가지 실험을 해 보았다. 첫 번째 실험은 최대 4단계까지 기억하는 인공개체와 그렇지 않은 인공개체들 중 각 단계별로 선택하여 실험해 보았다. 즉, 기억회로가 없는 인공개체와 기억단계가 4단계인 인공개체, 기억단계가 1단계인 인공개체(T1)와 기억단계가 4단계인 인공개체(T4), 기억단계가 2단계인 인공개체(T2)와 기억단계가 4단계인 인공개체, 그리고 기억단계가 3단계인 인공개체(T3)와 기억단계가 4단계인 인공개체 실험을 각각 50번씩 실험하였다. 각 실험의 종료 조건은 두 개체의 종 중 하나의 개체의 종이 멸종될 때까지 실험을 하여 마지막까지 살아남은 개체가 더 좋은 개체인 것으로 판단하였다. 두 번째 실험은 첫 번째 실험의 모든 인공개체 5종류를 동시에 실험하였다. 두 번째 실험도 첫 번째 실험과 마찬가지로 50번씩 반복 실험하여 마지막까지 남은 개체가 가장 잘 살아남은 개체가 가장 좋은 개체로 판단하였다. 두 실험에서 살아남은 인공개체는 그 실험환경에서 멸종한 인공개체보다 우월하게 진화되었다고 볼 수 있다. 그러나 인공개체의 진화가 확률적으로 이루어지기 때문에 실험 별로 특정한 인공개체가 무조건 다른 인공개체보다 더 빨리 진화할 수는 없다. 그러므로 우리는 통계적으로 어떤 인공개체가 더 우월한지를 알아보기 위하여 총 50번의 실험을 하였으며 각 실험별로 어떤 개체가 우월한지 기록하였다.

표 8는 첫 번째 실험결과를 보여준다. 표 8를 보면 대부분의 결과에서 T4가 우월함을 볼 수 있다. 하지만 T2와 T4의 실험에서는 약간의 차이로 T2가 더 우월하였다. 이는 이전 절에서 간단히 언급한 것처럼 마름모 먹이구조에서는 2단계의 기억단계를 갖는 인공개체가 먹이를 잘 찾아갈 수 있으면서 행위결정 로직프레임의 탐색공간도 크지 않아 빨리 진화하는 것으로 볼 수 있다. 표 9는 두 번째 실험결과를 보여준다. 첫 번째 실험결과와 비슷하게 T2가 가장 우월하였다. 하지만 두 번째 우월한 인공개체는 T1이었는데 이는 첫 번째 실험결과와는 다른 결과였다. 이 이유는 50번의 실험횟수가 통계적으로 충분하지 않음에서 기인한 것으로 보인다.

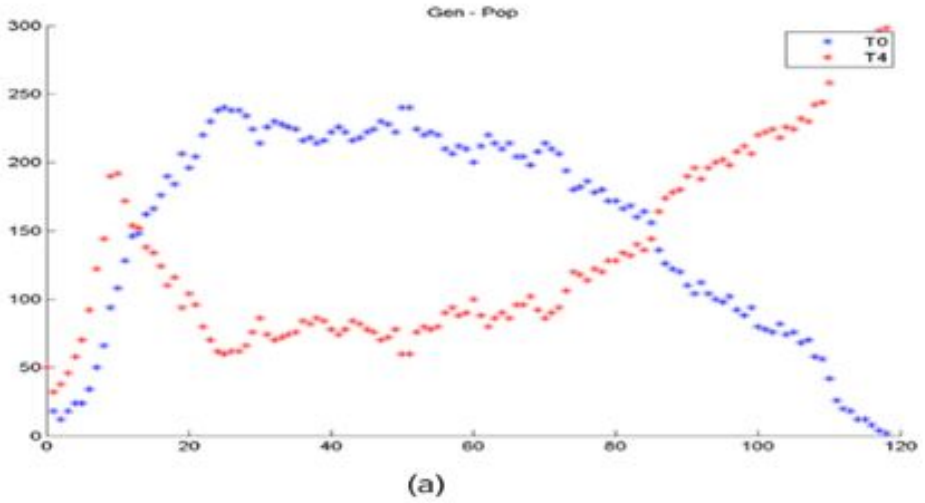
표 8. 서로 다른 기억단계를 갖는 두 인공개체의 우월성 비교

| Ta vs Tb | Ta 승 | Tb 승 | 최종승리 |
|---|---|---|---|
| T0 vs T4 | 15 | 35 | T4 |
| T1 vs T4 | 20 | 30 | T4 |
| T2 vs T4 | 27 | 23 | T2 |
| T3 vs T4 | 18 | 32 | T4 |

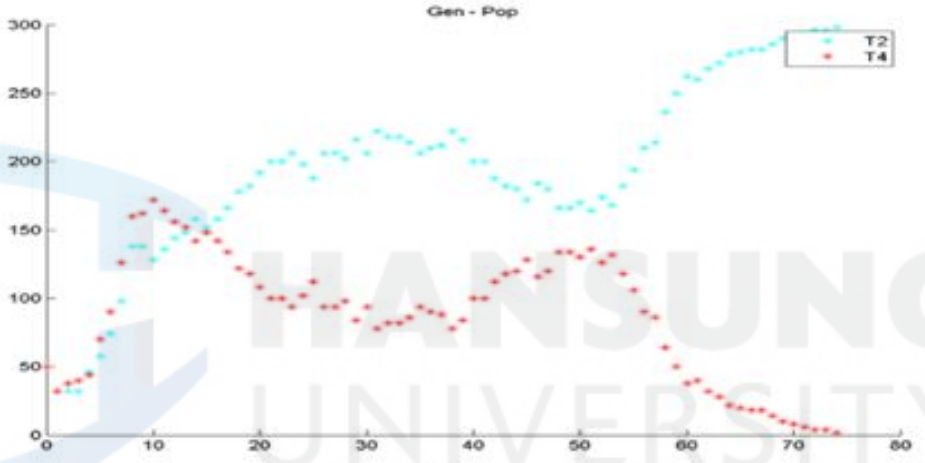
표 9. 모든 기억단계 동시 우월성 비교

| 개체종류 | 승리횟수 |
|---|---|
| T0 | 3 |
| T1 | 11 |
| T2 | 18 |
| T3 | 9 |
| T4 | 9 |

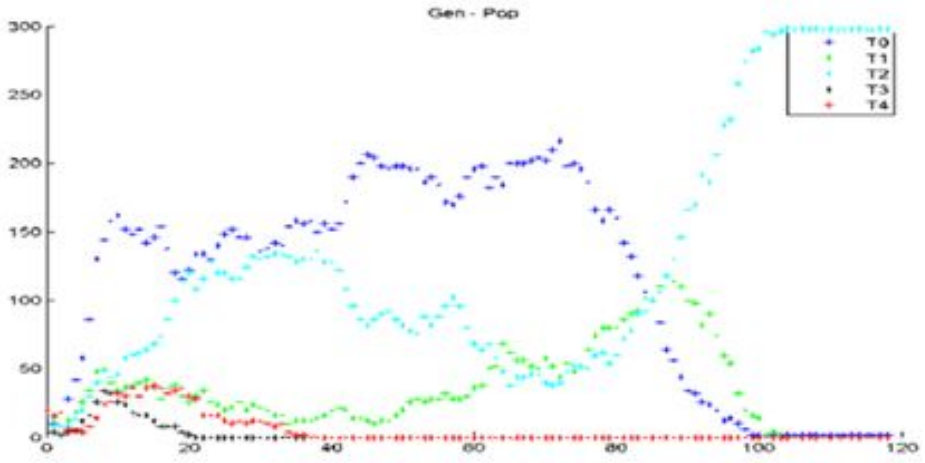
그림 21은 두 타입 별로 실험한 첫 번째 실험결과와 모든 개체타입을 동시에 진화시킨 두 번째 실험을 보여준다. 그림 21의 파란색은 T0를, 연두색은 T1을, 하늘색은 T2를, 검은색은 T3를, 붉은색은 T4를 나타낸다. 각 그림의 x축은 세대를 나타내며 y축은 세대 별 각 인공개체의 개수를 나타낸다. 그림 21(a)와 (b)는 첫 번째 실험에 해당하는 결과이고 그림(c)는 두 번째 실험에 해당하는 그림이다. 그림 21(a)는 T0와 T4의 실험결과이며 21(b)는 T2와 T4의 실험결과이다. 우리는 모든 타입에 대하여 실험하였으나 대표적으로 두 실험의 결과만 보여준다. 그림 21(a)를 보면 초반에는 T0와 T4 각각 50개에서 증가하다가 15세대 즈음에 각 종류별로 150개의 개체수가 된다. 이는 최대 개체수를 300개로 유지해주기 때문이다. T0 개체는 이후 87세대 즈음까지 T4 개체보다 개체수가 많으나 그 이후에 T4 개체에 역전 당한다. 이는 T4 개체가 다수의 세대동안 진화하면서 T0 보다 우월한 행동을 찾은 개체가 등장하기 때문으로 판단된다. 이후에는 이 우월한 T4 개체가 자식을 많이 번성하여 T0 개체를 멸종시킨다. 그림 21(b)는 T2 와




(a)




(b)




(c)

그림 21. 기억단계 별 실험 결과

T4의 경우의 실험결과이다. 이 실험을 보면 초기에는 T2의 개수가 더욱 많다가 50세대 근방에서는 T4의 성장률이 비교적 높아졌으나 T2 보다 우월한 내부로직을 찾지 못하여 결국 T2가 좋게 끝나는 경우를 볼 수 있다. T0 와 T4의 경우에서는 T0의 경우 과거행동을 기억하지 못하기 때문에 장기간의 세대에 걸쳐 진화를 하더라도 아주 좋은 행동을 찾아내기 쉽지 않다. 그러므로 장시간에 걸쳐 진화된 T4가 T0을 이기는 개체로 진화될 가능성이 높다. 그러나 T2와 T4 에서는 T2도 과거행동을 기억하기 때문에 지속적으로 더 좋은 행동을 찾아나갈 수 있고 이는 비록 T4가 좋은 행동을 찾아서 추격해 나가도 이보다 더 우월하게 진행할 수 있다. 그러나 이는 먹이구조에 따른 것으로 만약 먹이구조가 더 복잡하여 더 많은 단계의 과거 행동을 기억하는 것이 유리한 경우라고 하면 T0 와 T4 실험에서처럼 후반세대에서 T4가 T2보다 더 진화된 개체를 생성하여 T2를 멸종시킬 수도 있다.

그림 21(c)에 있는 두 번째 실험의 경우는 5종류의 인공개체를 동시에 진화시킨 실험 결과를 보여준다. 세대별로 각 인공개체의 개수는 그림 21(c)에 있다. 그림 21(c)에서 보면 세대 전반부와 중반부에서 모두 T0 가 우위에 있는데 이는 T0는 내부로직의 탐색공간이 작아서 상대적으로 빠르게 진화할 수 있기 때문이다. 반면 기억단계를 갖는 다른 인공개체수는 탐색공간이 기억이 없는 개체보다 큰데다 개체수가 많지 않아서 충분히 진화되기 어려운 상황이 된다. 첫 번째 실험에서는 두 개체 끼리의 실험이므로 기억회로를 갖는 개체도 충분히 많은 개체가 존재해서 진화를 비교적 많이 할 수 있었지만 두 번째 실험에서는 모든 타입의 개체로 실험하였기 때문에 각각의 개체타입의 개체수가 작다. 그러므로 진화가 많이 필요한 기억회로를 갖는 개체가 불리하다. 다만 T1 이나 T2의 개체는 기억회로를 갖지 않는 회로에 비하여 탐색공간이 많이 크지는 않으면서도 충분한 시간을 진화할 수 있기 때문에 중반부나 후반부로 가면서 경쟁력 있는 개체로 진화할 수 있다.

T2의 행동은 그림 22의 초록색 화살표 방향처럼 마름모 먹이구조 내부에서 루프를 형성하여 먹이를 찾아 먹도록 진화하였다. 하지만 T4의 행동은 그림 22의 파란색 화살표 방향처럼 먹이를 먹도록 진화하였다. T2의 경우에는 직진하다가 먹이를 만나면 하나의 사선으로 먹이가 존재한다는 것을 진화적으로 찾은 것으로

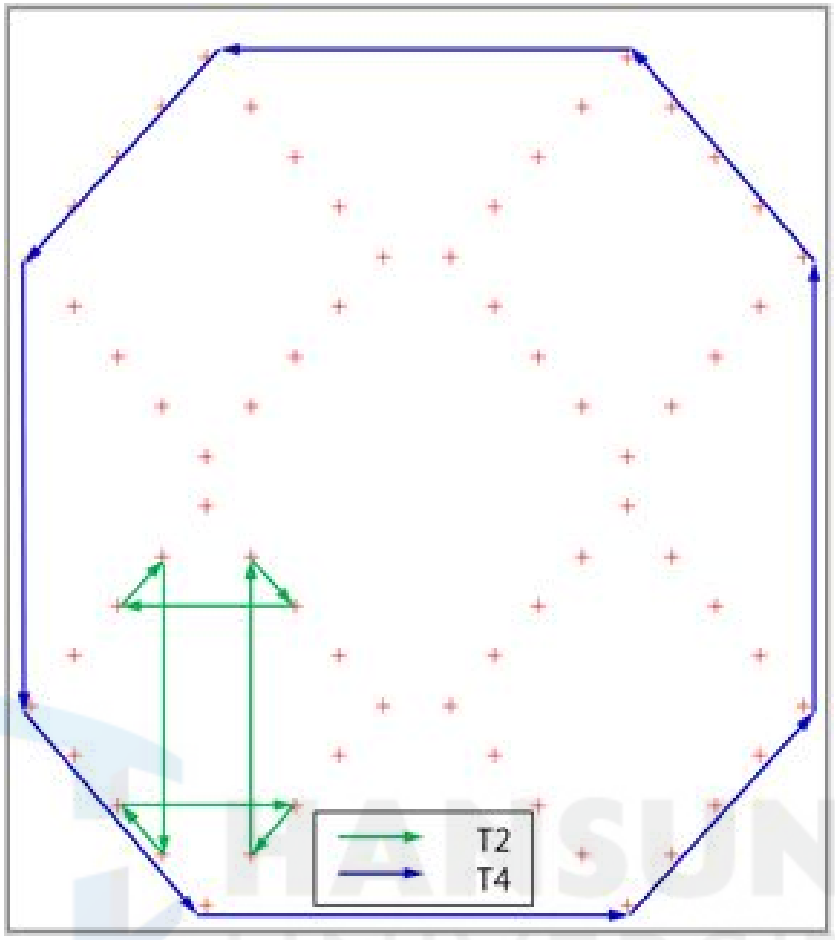



그림 22. T2와 T4의 행동 비교

보인다. 반면 T4의 경우에는 5개의 사선의 먹이를 모두 찾을 수 있도록 진화한 것이다. 이 예에서 보듯이 T2의 경우는 단순한 먹이패턴을 빨리 찾도록 진화할 수 있으나 T4의 경우에는 더 복잡한 먹이패턴을 찾을 수 있으나 진화에는 더 시간이 필요함을 알 수 있다. 그러므로 단순한 먹이패턴만 찾더라도 충분히 먹이를 먹을 수 있는 경우에는 T2가 더 유리할 수 있다. 더욱 자세한 개체의 행동은 다음 동영상을 참고하기 바란다.

• T2의 진화 후 행동

☞ http://itsys.hansung.ac.kr/dn/T2_Evolution.wmv

• T4의 진화 후 행동

☞ http://itsys.hansung.ac.kr/dn/T4_Evolution.wmv

# V. 결 론

셀 우주는 지구상에 존재하는 생물들이 자신의 환경에 적합한 형태로 진화하는 메커니즘을 생물학적 및 공학적 응용을 할 수 있도록 컴퓨터 상의 인공개체-먹이 시뮬레이션 프레임워크를 제안하였다. 셀 우주 상에 존재하는 인공개체는 주변의 먹이를 입력으로 받는 행위결정 로직프레임의 내부로직을 통해 행동 한다. 초기에는 이 내부로직이 임의적으로 형성되어 먹이를 잘 먹지 못한다. 하지만 세대가 흐르면서 번식하여 내부로직에 변이가 생기어 먹이를 잘 먹는 행동을 하는 인공개체들이 많이 살아남게 되며 세대가 지날수록 진화하게 된다. 우리는 셀 우주에서 제안하는 행위결정 로직프레임과 시뮬레이션 프레임워크의 동작성을 실험을 통해 검증하였으며 검증된 내용을 기반으로 다양한 먹이구조에서 인공개체가 어떻게 진화하는지 실험하였다. 또한 인공개체가 과거의 행동을 기억하여 되먹임 받아 입력으로 받는 기억회로가 있는 행위결정 로직프레임을 구현하였으며 이 때 인공개체가 어떻게 행동하는지 분석하여 보았다. 그리고 기억회로가 있는 인공개체와 없는 인공개체의 행동이 어느 것이 더 좋은지 비교해 보았고 또한 기억회로가 있는 인공개체 중 몇 단계의 행동을 기억하는 것이 가장 최적인지 실험을 통해 알아보았다.

행위결정 로직프레임의 동작성을 확인해 보기 위해 먼저 우리는 인공개체가 가질 수 있는 모든 먹이환경에 대해 인공개체의 최적의 행동을 파악하였다. 그리고 몬테카를로 방법을 통해 행위결정 로직프레임의 내부로직이 모든 먹이환경에 최적의 행동을 출력으로 내보낼 수 있는지 확인하여 보았다. 또한 최적의 행동을 기준으로 인공개체의 적합도를 평가해 시뮬레이션을 하여 보았다. 그 결과 충분히 먹이를 먹을 수 있는 내부로직이 존재함을 확인하였으며 적합도 평가를 기준으로 진화시킨 인공개체들이 모든 먹이 환경에 대해 먹이를 잘 먹도록 진화함을 확인 하였다. 그리고 적합도 평가를 빼고 실험하였을 때에도 인공개체가 먹이를 잘 먹도록 진화함을 확인 하였다.

하지만 기억회로가 없는 인공개체가 복잡한 먹이구조를 인식하는데 한계를 갖고 있음을 확인하여 다양한 먹이구조에서 기억회로를 갖는 인공개체를 진화시켜

보았다. 그 결과 동일한 먹이환경에 놓여있을 때 오로지 단순한 행동만 할 수 밖에 없었던 인공개체가 기억회로를 통해 다양한 조합의 행동이 가능해지며 멀리 떨어져 있는 먹이를 찾아 먹도록 진화함을 확인 하였다.
인공개체가 기억회로를 통해 다양한 먹이구조를 인식할 수 있었지만 내부로직의 탐색공간이 넓어져 진화에 영향을 주게 된다. 탐색공간의 영향을 확인하기 위하여 우리는 마름모형 먹이구조에서 최대 4개의 행동을 기억할 수 있는 인공개체끼리 실험해 보았다. 그 결과 많은 행동을 기억할수록 더욱 좋은 행동을 할 수 있지만 진화하는데 오랜 시간이 걸려서 적당히 기억하는 인공개체들에 의해 멸종되었음을 확인 하였다.

우리는 위의 실험들을 통해 셀 우주의 인공개체-먹이 모델이 하나의 시뮬레이션 프레임워크로서 충분한 구조를 가지고 있음을 확인할 수 있었다. 셀 우주를 통해 많은 학자들이 최적화 문제, 생물학 문제, 공학적 응용 등 다양한 분야에서 필요한 시뮬레이션용 프레임워크로서 사용되기를 기대한다.

# 참 고 문 헌

## 1. 국내문헌

정보선, 정성훈. (2014). Cellular Cosmos: 셀 수준의 진화 모델링 및 시뮬레이션 프레임워크.『대한전자공학회 학술대회 논문집』, 749-751.

정보선, 정성훈. (2015). 셀 수준의 진화 프레임워크를 통한 인공개체의 행동로직 진화.『한국지능시스템학회 논문지』, 25(1), 22-28.

## 2. 국외문헌


Baker, B. M. & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. Computers & Operations Research, 30(5), 787−800.

Darwin, C. (2009). The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life. W. F. Bynum (Ed.). AL Burt.

Davies, P. (2000). The fifth miracle: The search for the origin and meaning of life. Simon and Schuster.

Denton, M. & Scott, R. (1986). Evolution: a theory in crisis, pp.145. Bethesda, MA, USA: Adler & Adler.

Dorigo, M. & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. BioSystems, 43(2), 73−81.

Fan, W. Fox, E. A. Pathak, P. & Wu, H. (2004). The effects of fitness functions on genetic programming-based ranking discovery for Web search. Journal of the American Society for Information Science and Technology, 55(7), 628−636.

Flood, M. M. (1956). The traveling−salesman problem. Operations Research, 4(1), 61−75.

Jackson, W. (1996). Evolution: Fact Or Theory?. Apologetics Press.

Jarkko, K. (2013). Cellular Automata, University of Turku.

Jedlowski, P. (2001). Memory and Sociology Themes and Issues. Time & Society, 10(1), 29−44.

Kari, J. (2005). Theory of cellular automata: A survey. Theoretical computer science, 334(1), 3−33.

Langton, C. G. (1984). Self-reproduction in cellular automata. Physica D: Nonlinear Phenomena, 10(1-2), 135-144.

Langton, C. G. (1989). Artificial life. pp.1-48. Redwood City, CA: Addison-Wesley Publishing Company.

Langton, C. G. (1997). Artificial life: An overview. MIT Press.

Libelli, S. M. & Alba, P. (2000). Adaptive mutation in genetic algorithms. Soft computing, 4(2), 76-80.

Lin, W. Y. Lee, W. Y. & Hong, T. P. (2003). Adapting crossover and mutation rates in genetic algorithms. J. Inf. Sci. Eng., 19(5), 889-903.

Losos, J. (2013). What Is Evolution?. Princeton Guide to Evolution, 3-9.

MacKay, D. J. (1998). Introduction to monte carlo methods. In Learning in graphical models, pp.175-204. Springer Netherlands.

Mayr, E. (2001). What evolution is. Basic books.

Michalewicz, Z. Janikow, C. Z. & Krawczyk, J. B. (1992). A modified genetic algorithm for optimal control problems. Computers & Mathematics with Applications, 23(12), 83-94.

Minsky, M. (1961). Steps toward artificial intelligence. Proceedings of the IRE, 49(1), 8-30.

Mitchell, M. (1995). Genetic algorithms: An overview. Complexity, 1(1), 31-39.

Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

Müller, V. C. & Bostrom, N. (2014). Future progress in artificial intelligence: A survey of expert opinion. Fundamental Issues of Artificial Intelligence.

Noraini, M. R. & Geraghty, J. (2011). Genetic algorithm performance

with different selection strategies in solving TSP.

Ossimitz, G. & Mrotzek, M. (2008). The basics of system dynamics: discrete vs. continuous modelling of time. In Proceedings of the 26th International Conference of the System Dynamics Society.

Sarkar, P. (2000). A brief history of cellular automata. ACM Computing Surveys (CSUR), 32(1), 80-107.

Sherry, D. F. & Schacter, D. L. (1987). The evolution of multiple memory systems. Psychological review, 94(4), 439.

Staguhn, G. (2004). Find the blueprint of life-genetics and evolution of life by miracle and coincidence facets (Jang, H., Trans.). HaeNaMoo.

Torresen, J. (2004). An evolvable hardware tutorial. In Field Programmable Logic and Application, pp.821-830. Springer Berlin Heidelberg.

Whitley, D. (1994). A genetic algorithm tutorial. Statistics and computing, 4(2), 65-85.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution Vol. 1, pp.356-366. na.

Yang, S. (2007). Genetic algorithms with elitism-based immigrants for changing optimization problems. In Applications of Evolutionary Computing, pp. 627-636. Springer Berlin Heidelberg.

# ABSTRACT

## Cellular Cosmos : Architecture and Application of Cell-level Evolution Framework for Artificial Individuals

Jeong, Bo-Sun
Major in Information and Communication Engineering
Dept. of Information and Communication Engineering
The Graduate School
Hansung University

The Organisms on Earth have been evolved to fit into their nature and became to have current form. We propose an evolutionary framework, which can simulate the evolution mechanisms on computers, to analyze and understand the evolution and we have applied it to individual-prey model for the simulation. Also we have tested to see how the remembering of their past actions can result on their behaviors of evolution.

Cellular Cosmos consists of 2-dimensional cell space where individuals and prey can be placed. These individuals select their actions based on their behavior-selecting logic frame(BSLF). The BSLF is a system with input and output. Input is the condition of prey on individual's neighbor cells and output is the action. The output depends on BSLF's inner logic and this output causes an individual to act. Because this BSLF only has inputs of prey, it is not possible for an individual to act differently when it is on the same prey condition as past. But by adding a memory circuit to the BSLF

causes the individuals to act with variety.

Memory circuit is an add-on to the BSLF which can feedback individual's action from the past to current input. Because of this feedback, individual with memory circuit can act differently when on the same prey condition as before. An individual with memory circuit can evolve to eat more prey than the one without.

When the simulation starts, the individuals are created to have a random inner logic, so they are not able to eat many prey. But the individuals which can eat little more prey than others on the same generation will survive to the next generation and reproduce offsprings with mutated inner logic. As generations pass by, better individuals will survive and causes an evolution. We have tested to see which individuals can eat more prey.

What we propose on this thesis can be applied to simulated engineering problem that requires evolutionary simulations or to get step closer to biological evolutionary problems that presently remains unsolved.



keywords: artificial individual, prey, evolutions, framework, mutation, memory circuit, self-reproduction.