사물인터넷융복합논문지 Vol. 11, No. 6, pp. 153-162, 2025 https://doi.org/10.20465/KIOTS.2025.11.6.153

숫자 모으기 미로 게임을 위한 유전 알고리즘

이상욱*
목원대학교 게임소프트웨어공학과 교수

Genetic Algorithm for Number Collecting Maze Game

Sangwook Lee*
Professor, Department of Game Software Engineering, Mokwon University

요 약 본 숫자 모으기 미로 게임은 플레이어가 격자형 미로를 이동하며 특정 숫자를 수집하는 게임을 말한다. 숫자 모으기 미로 게임은 수리 및 인지 능력 향상을 위한 교육 및 퍼즐 설계 분야에 적용되어 왔으며, 탐욕 알고리즘, 빔 탐색, 초쿠다이 탐색 등의 알고리즘 성능 테스트를 위한 벤치마크 문제로서 다양한 연구에 활용되기도 하였다. 본 논문에서는 숫자 모으기 미로 게임에 유전 알고리즘을 적용하기 위한 설계 방법을 제안한다. 유전 알고리즘을 숫자 모으기 미로 게임에 적용하기 위한 해 표현법을 제시하고, 교차 연산, 돌연변이 연산 및 선택 연산 등과 같은 유전 연산 설계 방법을 제시한다. 간단한 유전 연산자를 적용한 유전 알고리즘은 탐욕 알고리즘 대비 우수한 결과를 보이나, 숫자 모으기 미로 게임에 특화된 빔 탐색 및 초쿠다이 탐색 알고리즘과 대비해서는 열등한 결과를 보인다. 이를 개선하기 위해, 탐욕적 개체를 혼합한 초기화, 엘리트 전략과 토너먼트 전략을 혼합한 선택 전략, 탐색 초기에는 높은 돌연변이 확률을 가지고 탐색 말기에는 낮은 돌연변이 확률을 가지는 적응형 돌연변이 등의 유전 알고리즘 강화 기법을 설계한다. 숫자 모으기 게임에 적용한 결과, 제안한 강화 유전 알고리즘이 빔 탐색 및 초쿠다이 탐색 알고리즘 대비 우수한 성능을 보여준다.

주제어 : 숫자 모으기 미로 게임, 빔 탐색, 유전 알고리즘, 복구 연산, 알고리즘 강화

Abstract The Number Collecting Maze game involves players navigating a grid-shaped maze to collect specific numbers. The Number Collecting Maze game has been applied in educational and puzzle design fields to enhance mathematical and cognitive abilities, and has also served as a benchmark problem for testing the performance of algorithms such as greedy algorithms, beam search, and Chokudai search. This paper proposes a design method for applying genetic algorithms to the Number Collecting Maze game. We present a solution representation for applying genetic algorithms to the Number Collecting Maze game and present design methods for genetic operations such as crossover, mutation, and selection. The genetic algorithm, employing simple genetic operators, outperform the greedy algorithm but underperform the beam search and Chokudai search algorithms, which are specialized for the Number Collecting Maze game. To improve this, we design genetic algorithm enhancement techniques such as initialization mixed with greedy individuals, selection strategy mixed with elite strategy and tournament strategy, and adaptive mutation that has high mutation probability in the early stage of search and low mutation probability in the late stage of search. When applied to a number-collecting game, the proposed reinforced genetic algorithm shows superior performance compared to the beam search and Chokudai search algorithms.

Key Words : Number Collecting Maze Game, Beam Search, Genetic Algorithms, Repair Operator, Algorithm Enhancement

*교신저자 : 이상욱(slee@mokwon.ac.kr)
접수일 2025년 11월 25일 수정일 2025년 12월 14일 심사완료일 2025년 12월 17일

1. 서론

숫자 모으기 미로(Number Collecting Maze) 게임은 격자형(Grid-based) 미로나 그래프 구조의 환경에서 에이전트가 제한된 시간 혹은 이동 횟수 내에 특정 숫자(혹은 동전·보물과 같은 수집 아이템)를 최대한 많이 회수하는 경로를 탐색하는 문제로 정의된다[1]. 겉보기에는 단순한 퍼즐 형태이지만, 실제로는 경로 최적화, 순서 제약, 제한적 자원(시간·행동 수), 부분 관측, 그리고 희소 보상(sparse reward) 등 다양한 분야에서 연구되고 있으며, 교육공학 분야에서는 수리/논리 능력 향상을 위한 학습 도구로, 게임 디자인에서는 난이도 조절 및 레벨 생성 연구의 실험 환경으로, 인공지능 분야에서는 Gridworld 기반의 표준 벤치마크로 폭넓게 활용되고 있다.

1.1 교육·퍼즐 관점 선행연구

교육 연구 분야에서는 미로 기반 수학·논리 게임이 학습자의 주의력, 계산력, 문제 해결 능력 향상에 기여하는지 여부를 검증하는 연구가 수행되어 왔다[2]. 최근에는 학습자 모델링을 활용한 개인화된 미로 제공 기법과 적응형 난이도 조절(adaptive difficulty)이 기법이 제안되기도 하였다.

1.2 게임 디자인 관점 선행연구

게임 디자인 연구는 레벨 구조, 보상 체계, 몰입도(engagement)가 플레이 경험에 미치는 영향을 분석하며, 프로시저럴 콘텐츠 생성(PCG)을 통해 자동화된 레벨 구성 기법을 제안하고 있다. Zhang 등은 강화학습 기반 환경 설계 기법을 활용하여 도전도와 전략적 다양성을 조절하는 접근을 제시하였으며[3], Earle 등은 뉴럴 셀룰러 오토마타 기반의 미로·퍼즐 레벨 생성 기법을 제안하였다[4]. 이러한 연구는 레벨·보상 설계가 사용자의 몰입도, 난이도 인식, 학습 효과에 핵심적인 영향을 미친다는 점을 보여준다.

1.3 강화학습(Gridworld) 분야 선행연구

강화학습 분야에서는 Gridworld 수집 과제를 통해 희소 보상, 탐험 전략, 일반화 문제 등을 검증한다. pseudocount 기반 탐험 보정[5], 엔트로피 정규화[6]와 같은 다양한 기법이 수집형 Gridworld 환경에서 평가되며, 새로운 레벨에서의 일반화(extrapolation) 문제도 분석되고 있다.

1.4 미로 생성 알고리즘과 난이도 설계 선행연구

퍼즐 이론 및 게임 연구에서는 DFS 백트래킹, Prim, Wilson, Aldous-Broder 등 다양한 미로 생성 알고리즘이 산출하는 지형적 특징과 난이도 특성을 분석한다. 알고리즘 선택은 경로 구조, 분기 수준, 탐색 난이도에 영향을 미치며, 일부 연구는 난이도 제어 가능성 및 교육적 활용 가능성을 평가하기 위한 알고리즘을 제안하였다[7].

1.5 유전 알고리즘 적용 사례

숫자 모으기 미로 게임에 유전 알고리즘(Genetic Algorithm, GA)이 직접적으로 적용된 사례는 드물지만, 미로 탐색 및 경로 계획 문제에 GA를 적용한 연구는 다수 보고되어 왔다. GA를 이용해 이동 경로를 인코딩하고 목적지 도달성 및 경로 효율성 등을 적합도(fitness)로 평가하는 접근이 대표적이며, 이러한 접근은 경로 길이나 도달 확률 측면에서 GA가 효율적이다는 결과를 보여주었다[8]. 또한 GA 기반 연구들은 단순 출구 찾기뿐만 아니라, 레벨 생성(maze-instance generation), NPC 행동 진화 문제에도 적용되어 왔다[9, 10]. 그리고, GA를 활용하여 경로 시퀀스 최적화, 전략 다양성 확보, 적응형 난이도 설계 등에 응용하는 연구도 진행되었다[11].

본 연구에서는 숫자 모으기 미로 게임에서 수집하는 숫자를 최대화하는 경로를 탐색하기 위한 효율적인 유전 알고리즘을 설계하고자 한다. 먼저, 본 연구에서 연구할 숫자 모으기 미로 게임에 대해 살펴보고 수집한 숫자를 최대화하기 위해 기존에 적용된 알고리즘을 알아본다. 그 후, 숫자 모으기 미로 게임에 적용할 단순 유전 알고리즘을 설계하고 성능을 확인해 본다. 마지막으로, 단순 유전 알고리즘의 성능을 향상시키기 위한 강화 기법을 설계하고, 향상된 성능을 확인해 본다.

2. 숫자 모으기 미로 게임

숫자 모으기 미로 게임은 미로를 탐험하며 숫자를 모으는 형태의 퍼즐 게임을 말한다. 일반적인 숫자 모으기 미로 게임의 구조와 규칙은 다음과 같다.

2.1 게임의 기본 구성

2.1.1 미로 (Map)

- 길과 벽이 있는 형태의 미로가 주어짐
- 플레이어는 정해진 시작점에서 출발 가능

2.1.2 숫자(Goal Items)

- 미로 칸에 0~9의 숫자 또는 특정 값이 있는 칸들이 존재함
- 숫자를 모두 모으거나, 특정 순서/조건에 맞게 모으는 것이 게임의 목표임

2.1.3 플레이어 이동

- 일반적으로 상/하/좌/우로 한 칸씩 이동할 수 있음
- 벽(장애물)은 통과할 수 없음

2.2 대표적인 규칙과 목표

게임의 특성에 따라 규칙과 목표가 다를 수 있지만, 일반적으로 다음 규칙 중 하나 이상이 적용된다.

2.2.1 주어진 순서대로 숫자 모으기

- 예) 1 → 2 → 3 → 4 순서로 수집
- 순서를 어기면 실패

2.2.2 합 만들기

- 예) 총합이 30이 되도록 숫자 모으기
- 제한된 횟수 또는 경로 안에서 합 조건 만족

2.2.3 최단 거리로 모두 수집하기

- 예) 모든 숫자를 최소 이동 횟수로 모으기
- 경로 최적화 문제(조합/그래프 문제)에 가까움

2.2.4 제한 시간 또는 이동 횟수 제한

- 예) 제한된 턴 수 안에서 최대 숫자 합 모으기
- 경로 최적화 문제(조합/그래프 문제)에 가까움

2.3 본 연구에 적용하는 숫자 모으기 게임

본 연구에서는 2.2.4와 같은 제한된 이동 횟수 안에서 수집한 숫자의 합이 최대가 되는 것을 목표로 하는 게임을 다룬다. 이 게임은 수집한 숫자의 합이 최대가 되는 플레이어의 최적 경로를 찾는 경로 최적화 문제로 볼 수 있다.

2.3.1 게임 규칙

표 1은 본 연구에서 적용하는 숫자 모으기 게임의 규칙에 대해 설명하고 있다[1].

〈Table 1〉 Game rule

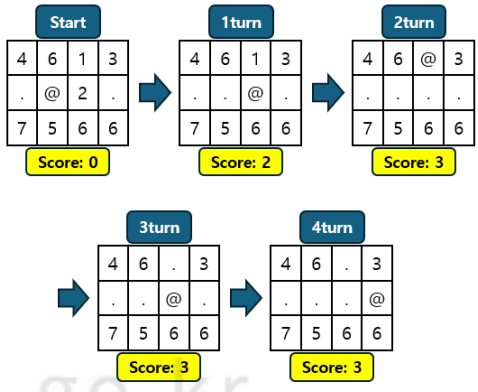
	Description
Player Objectives	• Earn record scores until the end of the game
Number of players	• 1 person
Player action timing	• Once per turn
Actions that the player can perform	• Each turn, move the character 1 space up, down, left, or right.
Game End Conditions	• Reach a set number of turns
etc	• The character's initial location is random • The character gains points for the location he or she moves to, and the points for that location are deleted

2.3.2 게임 동작 예

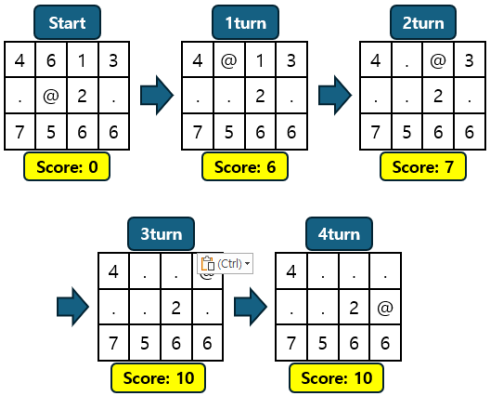
게임의 규칙이 표 1과 같고, 게임 미로의 초기 상태가 그림 1과 같다고 가정하자.

[Fig. 1] Initial state of game

종료 조건에 해당하는 턴 수를 4턴으로 설정하였을 때, 그림 1과 그림 2는 게임에서 진행될 수 있는 동작의 예를 보여주고 있다. 그림 1과 그림 2는 최종적으로 동일한 지점에 도착하였으나, 그림 1은 우→상→하→우 순으로 이동하면서 3점을, 그림 2는 상→우→우→하 순으로 이동하면서 10점을 획득한다는 것을 보여주고 있다.



[Fig. 2] Example #1 of game with 4 turn



[Fig. 3] Example #2 of game with 4 turn

2.4 숫자 모으기 게임을 위한 알고리즘

본 연구에서 사용하는 숫자 모으기 게임을 위한 기본 알고리즘으로는 무작위 행동 선택(Random Selection, RS), 탐욕 탐색(Greedy Search, GS), 빔 탐색(Beam Search, BS), 초쿠다이 탐색(Chokudai Search, CS) 등이 있다[1].

2.4.1 무작위 행동 선택

무작위 행동 선택은 다음 턴의 행동을 무작위로 선택하는 방법이다. 구현은 간단하나 무작위로 이동하는 만큼 성능을 보장할 수 없다.

2.4.2 탐욕 탐색

탐욕 탐색은 다음 턴에 발생 가능한 모든 결과 중에서 가장 평가가 높은 결과를 내는 행동을 선택하는 방법이다. 구현이 간단하고 일정 수준 이상의 효과를 기대할 수 있어 널리 사용되고 있다.

2.4.3 빔 탐색

탐욕 알고리즘은 1턴 이후만 고려하지만, 빔 탐색은 일정 턴 이후까지 고려하여 최선의 결과는 내는 행동을 선택하는 방법으로 높은 수준의 효과를 기대할 수 있다.

2.4.4 초쿠다이 탐색

빔 탐색은 일정 이상의 턴 이후까지 탐색하더라도 원점으로 돌아가면 같은 국면만 계속 탐색하게 되는 다양성이 부족한 단점이 있다. 초쿠다이 탐색은 너비가 좁은 빔 탐색을 반복해서 다양성을 확보하는 기법으로 빔 탐색보다 효율적이다.

지금까지 숫자 모으기 미로 게임의 규칙, 동작 예 및 해결을 위한 기본 알고리즘에 대해 살펴보았다. 다음 장에서는 숫자 모으기 미로 게임에 적용할 유전 알고리즘 설계에 대해 살펴본다.

3. 유전 알고리즘 설계

유전 알고리즘(Genetic Algorithm, GA)은 Holland에 의해 체계화된 진화계산(Evolutionary Computation)의 대표적 기법으로, 생물학적 진화 과정에서 관찰되는 자연선택(natural selection)과 유전적 변이(genetic variation) 원리를 모방하여 만들어진 최적화 문제를 해결하기 위한 방법론이다[12]. GA는 명시적인 도함수 정보가 없거나 목적 함수의 형태가 복잡한 경우에도 적용 가능하며, 조합 최적화, 경로 탐색, 파라미터 최적화, 게임 인공지능 등 다양한 분야에서 활용되고 있다[13].

3.1 알고리즘 개념

GA는 문제의 잠재적 해(candidate solution)를 염색체(chromosome) 혹은 개체(individual)로 표현하고, 이들을 집합 형태인 개체군(population)으로 관리한다. 각 개체의 성능은 적합도 함수(fitness function)로 평가되며, 높은 적합도를 갖는 개체가 다음 세대에 선택될 확률이 증가한다. 선택(selection), 교차(crossover), 돌연변이(mutation)와 같은 유전 연산이 반복되면서 개체군은 점차 더 우수한 해를 포함하도록 진화한다.

3.2 주요 구성 요소

3.2.1 해 표현(Encoding)

해는 일반적으로 비트열(binary string), 실수 벡터(real-valued vector), 경로 시퀀스(sequence of actions) 등 문제에 적합한 형태로 인코딩된다. 예를 들어, 경로 계획 문제에서는 상·하·좌·우 이동 명령을 문자열로 표현할 수 있다.

3.2.2 적합도 함수(Fitness Function)

적합도 함수는 각 해의 품질을 정량적으로 평가하며, GA의 성능을 결정하는 핵심 요소이다. 예를 들어, 경로 최적화 문제에서는 이동 거리, 목표 도달 여부, 보상 수집량 등을 포함할 수 있다.

3.2.3 선택(Selection)

선택 단계는 높은 적합도를 가진 개체가 더 많이 번식하도록 하는 과정이다. 룰렛휠 선택(roulette wheel), 토너먼트 선택(tournament), 순위 기반 선택(rank-based) 등이 널리 사용된다[13].

3.2.4 교차(Crossover)

교차는 두 부모 해의 유전 정보를 조합해 자손(offspring)을 생성하는 연산이다. 단일점(one-point), 다중점(multi-point), 균등 교차(uniform crossover) 방식이 대표적이다[13]. 교차 과정은 탐색 공간을 넓히고, 기존 해의 장점을 재조합하여 새로운 해를 생성한다.

3.2.5 돌연변이(Mutation)

돌연변이는 개체의 일부 유전자를 무작위로 변형하여 탐색의 다양성을 유지하고 지역 최적해(local optimum)에 빠지는 것을 방지한다. 변이율은 일반적으로 낮게 설정된다.

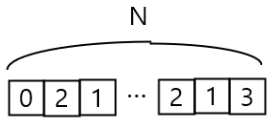
3.3 알고리즘 흐름

1) 초기화(Initialization) : 초기 개체군을 무작위로 생성한다.
2) 평가(Evaluation) : 각 개체의 적합도를 계산한다.
3) 유전 연산(Genetic operation) : 선택, 교차, 돌연변이 연산을 수행하여 새로운 개체군을 생성한다.
4) 평가(Evaluation) : 새로운 세대에서 다시 적합도를 평가한다.
5) 종료 조건(Terminal Condition) : 세대 수, 적합도 수준, 수렴 수준 등을 종료 조건을 설정하고 충족할 때까지 3), 4) 과정을 반복한다.

3.4 숫자 모으기 미로 게임을 위한 단순 GA 설계

3.4.1 해 표현

숫자 모으기 미로 게임에서는 플레이어가 정해진 종료 턴 수에 도달할 때까지, 각 턴 마다 상, 하, 좌, 우 4가지 방향 중 하나의 방향을 선택하여 이동한다. 여기서, 각 턴을 유전자로 표현하고, 유전자는 0, 1, 2, 3의 값을 가질 수 있으며, '0'은 '상', '1'은 '하', '2'는 '좌', '3'은 '우'로 해당 턴에서 플레이어가 이동할 방향을 표현하는 것으로 설계하였다. 그림 4는 종료 턴 수가 N개일 때의 해 표현을 나타낸다.



[Fig. 4] Chromosome representation

3.4.2 적합도 함수

개체(해)는 표현된 유전자 값에 따라 숫자 모으기 미로 게임판 위에서 순차적으로 이동하며, 획득한 점수의 총합을 적합도 값으로 사용한다. 여기서, 적합도 값은 클수록 우수하므로 최대화하는 것을 목표로 한다.

3.4.3 유전 연산

선택, 교차, 돌연변이의 유전 연산을 다음과 같이 가장 널리 사용되는 토너먼트 선택, 단일점 교차 및 돌연변이를 적용하였다.

1) 선택
- 토너먼트 선택(Tournament selection) 사용
- 무작위로 2개의 개체를 선택하고, 더 우수한 적합도를 가진 개체를 부모로 선택

2) 교차
- 단일점 교차 (One-point crossover) 적용
- 두 부모 개체의 각 유전자의 값을 일정 확률로 교환하여 새로운 자손 생성

3) 돌연변이
- 각 개체는 일정 확률로 돌연변이 수행
- 돌연변이가 발생하면 무작위로 선택된 하나의 유전자를 무작위로 선택된 방향 값으로 변경

3.4.4 복구

숫자 모으기 미로 게임에서는 그림 5와 같이 플레이어의 위치가 게임판에서 모서리에 위치하는 경우, 다음 턴에 이동 가능한 방향에 제한이 있을 수 있다. 이러한 문제를 해결하기 위해 다음과 같은 3가지 복구(Repair) 알고리즘을 설계하였다.

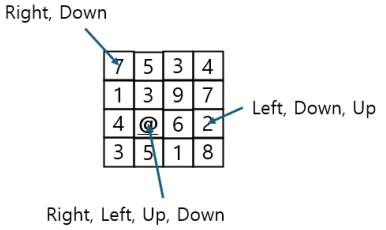
1) 인근 방향으로 교정 (Repair 1, r1)
- 개체를 평가하는 과정에서 불가능한 방향으로의 이동이 발생하는 경우 복구 수행
- 가능한 방향 중에서 현재 방향과 가장 가까운 (숫자

기준으로 인접한) 방향으로 변경

2) 불가능한 방향 가진 개체 제거 (Repair 2, r2)
- 개체를 평가하는 과정에서 불가능한 방향으로의 이동이 발생하는 경우 복구 수행
- 해당 개체의 평가 점수를 최하로 조정하여 선택받지 못하게 함

3) 무작위로 선택된 방향으로 교정 (Repair 3, r3)
- 개체 유전 연산 직후 불가능한 방향으로의 이동 발생 여부 확인하여, 있는 경우 복구 수행
- 가능한 방향 중에서 무작위로 선택된 방향으로 변경

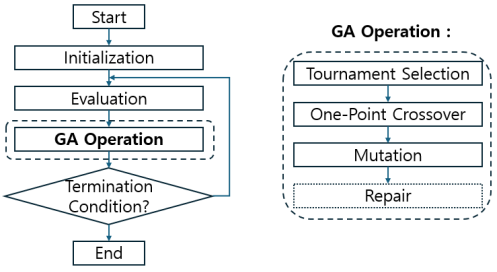
3가지 복구 방법 중, 1), 2)는 평가 도중에 진행하여, 별도 반복문을 통한 계산 비용 증가가 없으나, 3)은 돌연변이 수행 후 별도의 반복문을 사용하여 복구를 수행하므로, 계산 비용 증가가 발생한다.



[Fig. 5] Restrictions of player movement

3.4.5 알고리즘 흐름도

단순 GA를 흐름도로 표현하면 그림 6과 같다. 여기서, 별도로 표시된 복구(Repair) 과정은 3.4.4에서 언급한 복구 방법 중 3)번 복구 방법일 때만 적용되며, 1), 2)번의 복구 방법은 평가(Evaluation) 과정에 포함된다.



[Fig. 6] Simple GA flowchart

3.5 실험

본 실험에서는 본 장에서 설계한 단순 GA와 3가지 복구 방법의 성능을 확인해 본다. 먼저, 3가지 복구 방법에 대한 성능 비교를 통해, 효율적인 복구 방법을 확인한다. 그 후, 단순 GA의 성능을 확인하기 위해 숫자 모으기 미로 게임을 위한 기본 알고리즘(RS, GS, BS, CS)의 성능과 비교해 본다.

3.5.1 실험 환경 및 알고리즘 변수 설정

1) 실험 환경

본 실험에 사용된 컴퓨팅 환경은 AMD Ryzen 9 7900 12-Core Processor, 64GB Memory, window 10 64bit 운영체제이다. 개발 환경으로는 Visual Studio 2022를 사용하였으며, C++ 언어를 활용하여 설계한 알고리즘을 구현하였다. 또한, 시간 효율적인 GA 실행을 위해, CPU 병렬처리를 위한 OpenMP 라이브러리를 활용하였다. 실험에 사용된 숫자 모으기 미로 게임은 가로 × 세로의 크기가 10 × 10, 20 × 20, 30 × 30인 3가지 게임 판을 사용하였다. 게임 판 위에 먼저 플레이어를 무작위 위치에 배치하였으며, 그 후 플레이어가 없는 위치에 0~9 사이의 숫자를 무작위로 선택하여 배치하였다.

2) 알고리즘 변수 설정

본 실험 사용한 알고리즘 변수 설정은 다음과 같다.
- 개체 길이 : 종료 턴 수를 의미하며, [5, 50] 사이의 값을 5 간격으로 설정하여 총 10개의 개체 길이를 설정함
- 개체군 수 : 개체 길이가 5일 때 200, 개체 길이가 늘어나면 이에 비례하여 개체군 수 증가
- 세대 수 : 개체 길이가 5일 때 100, 개체 길이가 늘어나면 이에 비례하여 세대 수 증가
- 교차 확률 : 0.7, 돌연변이 확률 = 0.1

3.5.2 복구 방법 성능 비교

앞선 절에서 설명한 3가지 r1, r2, r3 복구 방법을 각각 적용한 GA_r1, GA_r2, GA_r3 알고리즘을 3가지 크기의 게임 판에 실험한 비교 결과는 각각 표 2, 표 3, 표 4와 같다. 각 개체 길이 별로 실험을 10번 반복하여 평균한 값을 제시한다.

〈Table 2〉 Comparison results of three repair methods (10 × 10 problem)

Length of individual	GA_r1	GA_r2	GA_r3
5	**32**	**32**	**32**
10	**69.5**	**69.5**	**69.5**
15	**104.3**	103.9	103.9
20	**135.8**	131.2	133.4
25	**165.3**	161.4	164.6
30	193.5	192.1	**194**
35	222.2	220.2	**227.4**
40	**255.3**	248.2	251.7
45	278	276.7	**279.7**
50	**304.1**	296.4	300.9

〈Table 3〉 Comparison results of three repair methods (20 × 20 problem)

Length of individual	GA_r1	GA_r2	GA_r3
5	**40**	**40**	**40**
10	75.5	**75.6**	74.8
15	**105.5**	104.7	104.1
20	134.6	133.3	**135.6**
25	**169.5**	165.8	164.3
30	**203.3**	202.6	199.7
35	**236.4**	227.9	233.6
40	267	**268.5**	266.9
45	**296.7**	295.6	295.6
50	**328.4**	326.9	328.3

〈Table 4〉 Comparison results of three repair methods (30 × 30 problem)

Length of individual	GA_r1	GA_r2	GA_r3
5	37.8	38.6	**38.8**
10	69.8	69.6	**70.3**
15	**102.9**	101.3	101.6
20	136	136.7	**137.1**
25	**168.9**	167.7	168.5
30	**203**	199.1	202.1
35	**235.1**	229	232.6
40	**265.4**	261.1	264.2
45	**298.1**	293	295.5
50	**330.4**	328.1	326

실험 결과, 평가 과정에서 불가능한 방향이 발생할 경우, 해당 방향의 인근 방향으로 조정하는 r1 복구 방법의 성능이 우수한 것으로 나타났다. 이는 다른 두 가지 방법에 비해, 유전자의 특성 변화를 최소화하면서 복구를 수행하기 때문인 것으로 사료된다.

3.5.3 GA_r1 vs (RS, GS, BS, CS)

앞선 절에서 가장 우수한 성능을 나타낸 GA_r1과 숫자 모으기 미로 게임을 위한 기본 알고리즘들(RS, GS, BS, CS)과의 성능을 비교하기 위해 BS의 빔 너비는 5, 빔 깊이는 개체 길이로 설정하고, CS의 빔 너비는 3, 빔 깊이는 개체 길이로 설정한다. 표 5, 표 6, 표 7은 3가지 크기의 게임 판에서 실험한 성능 비교 결과를 나타내고 있다. 앞선 실험과 마찬가지로 각 개체 길이 별로 실험을 10번 반복하여 평균한 값을 제시한다. 실험 결과, 단순 GA(GA_r1)은 무작위 행동 선택(RS) 및 탐욕 서치(GS) 대비 우수한 성능을 보였으나, 빔 탐색(BS) 및 초쿠다이 탐색(CS) 대비 열등한 성능을 보여준다.

〈Table 5〉 GA_r1 vs (RS, GS, BS, CS) (10 × 10 problem)

Length of individual	GA_r1	RS	GS	BS	CS
5	**32**	17.7	**32**	**32**	**32**
10	69.5	26.5	63	**70**	**70**
15	**104.3**	42.9	99	99	103
20	**135.8**	44.5	124	128	132
25	**165.3**	56.1	128	156	161
30	193.5	74.4	128	189	**194**
35	222.2	80.7	128	226	**230**
40	255.3	86.8	128	256	**259**
45	278	100.8	128	273	**273**
50	**304.1**	112.8	128	298	301

〈Table 6〉 Comparison results of three repair methods (20 × 20 problem)

Length of individual	GA_r1	RS	GS	BS	CS
5	**40**	18.4	**40**	37	**40**
10	75.5	26.7	**76**	**76**	**76**
15	105.5	40.8	**111**	**111**	**111**
20	134.6	49.8	**140**	**140**	**140**
25	169.5	57.1	172	172	**177**
30	203.3	70.9	202	210	**215**
35	236.4	74.3	229	248	**250**
40	267	92.3	256	276	**283**
45	296.7	99.6	284	298	**309**
50	328.4	112.7	313	334	**338**

〈Table 7〉 Comparison results of three repair methods (30 × 30 problem)

Length of individual	GA_r1	RS	GS	BS	CS
5	37.8	20.9	35	37	**39**
10	69.8	32.4	67	68	**70**
15	102.9	42.2	99	**104**	**104**
20	136	50.1	131	**141**	**141**
25	**168.9**	60.8	156	167	167
30	203	70.9	194	**204**	**204**
35	235.1	82.1	215	233	**236**
40	265.4	97.1	244	269	**269**
45	298.1	98.3	271	306	**306**
50	330.4	110.6	298	316	**340**

다음 장에서는 유전 알고리즘이 BS 및 CS 대비 우수한 성능을 보일 수 있도록 유전 알고리즘을 강화하기 위한 기법을 소개한다.

4. 강화 유전 알고리즘 설계

이전 장에서 구현한 단순 GA는 초기 개체군의 품질, 교차 방식, 돌연변이 확률, 선택 방법에 민감하여 탐색 효율성이 떨어질 수 있다. 특히 빔 탐색이나 초쿠다이 탐색과 같은 탐색기법은 깊이 기반의 국소적 탐색에서 강점을 가지므로, GA가 이들보다 성능 우위를 가지려면 초기화·교배·변이·선택 과정 전반을 보강해야 한다[14, 15]. 본 장에서는 이전 장에서 구현한 단순 GA의 성능을 향상시키기 위한 4가지 강화 기법을 제안하고, 실험을 통해 성능을 확인한다.

4.1 강화 기법

4.1.1 초기화 강화

단순 GA에서 적용한 완전 무작위 초기화는 다양성은 높지만 탐색 초기에 유망한 해를 포함하지 못할 가능성이 높다. 이를 보완하기 위해 탐욕 탐색의 결과로 도출된 개체를 초기화에서 개체군에 포함시키는 강화 기법을 제안한다. 다만, 모든 개체를 탐욕 탐색의 결과만으로 사용하면 다양성이 무너지므로, 완전 무작위로 만든 개체와 반반 비율로 혼합한 하이브리드 초기화로 초기 개체군을 형성한다. 이를 통해 초기 탐색 공간에 광역 탐색성(diversity)과 일부 고품질 시드(elite seeds)를 동시에 확보한다.

4.1.2 교차 개선

단순 GA에서 적용한 단일점 교차는 염색체를 한 지점에서만 절단하기 때문에, 염색체의 앞/뒤 구조가 지나치게 연동된 상태가 되어, 일부 중요한 유전자 블록(building block)이 앞쪽/뒤쪽의 구조적 위치에 묶여 있어서 해체되기 쉬운 단점이 있다. 이를 보완하기 위해 2점 교차를 적용하여, 중요한 유전자 블럭이 파괴되지 않고 부모에서 자식으로 안전하게 전달될 가능성을 높인다[13].

4.1.3 적응형 돌연변이율

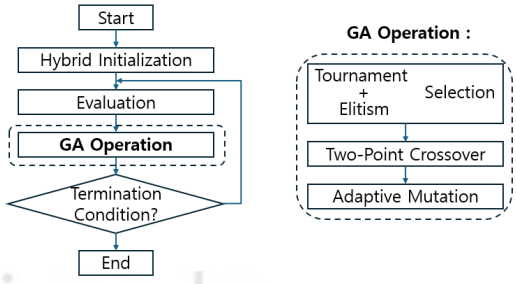
단순 GA에서 적용한 고정형 돌연변이율은 탐색 초기 및 후기 모두에 최적이 아닐 수 있다. 일반적으로, 탐색 초반엔 다양성을 위해 높은 돌연변이율이 유리하며, 후반엔 탐색 안정화를 위해 낮은 돌연변이율이 바람직하다. 이러한 특성을 반영하여 탐색 초기에 다소 높은 돌연변이율을 설정하고, 세대 수가 증가할수록 돌연변이율이 선형적으로 감소하는 적용형 돌연변이율을 적용한다.

4.1.4 선택 전략 강화

단순 GA에서 적용한 토너먼트 선택은 다양성은 보장하지만, 무작위 특성으로 인해 우수한 개체가 살아남지 못할 가능성이 있다. 이를 보완하기 위해 상위 20% 개체는 다음 세대 개체군으로 포함하고, 나머지 다음 세대 선택은 토너먼트 선택 전략으로 수행하는 하이브리드 선택 전략을 적용한다. 이를 통해, 좋은 유전자가 더 많이 전달됨과 동시에 다양성도 유지할 수 있다.

4.1.5 강화 유전 알고리즘 흐름도

앞선 절에서 기술한 4가지 강화 기법을 적용한 강화 유전 알고리즘(Enhanced Genetic Algorithm, Enhanced GA)을 흐름도로 표현하면 그림 7과 같다.



[Fig. 7] Enhanced GA flowchart

4.2 실험

본 실험에서는 본 장에서 설계한 강화 GA의 성능을 확인하기 위해 3.5 실험에서 확인된 최고 성능과 비교 분석해 본다.

4.2.1 실험 환경 및 알고리즘 변수 설정

1) 실험 환경

본 실험에 사용된 컴퓨팅 환경은 3.5.1에서 기술한 단순 GA 실험 환경과 동일하다.

2) 알고리즘 변수 설정

본 실험 사용한 알고리즘 변수 설정은 '개체 길이', '개체군 수', '세대 수', '교차 확률'은 3.5.1에서 기술한 단순 GA 알고리즘 변수 설정과 동일하다. 다만, '돌연변이 확률'은 세대가 0일 때 0.3으로 설정되고, 종료 세대에 이르면 0.05로 설정되며, 그 사이 세대에서의 값은 선형적으로 결정된다.

4.2.2 강화 GA 성능 실험 결과

3.5 절 표5, 표6, 표7에서 확인된 최고 성능(Best Performance Before)과 강화 GA 성능을 비교한 결과를 표 8, 표, 9, 표 10에서 보여주고 있다. 각 개체 길이별로 실험을 10번 반복하여 평균한 값을 제시한다.

실험 결과, 20 × 20의 개체 길이 45인 경우 및 30 × 30의 개체 길이 45인 경우를 제외하고는 기존 최고 성능 대비 강화 GA 기법이 모두 우수한 성능을 보여준다.

〈Table 8〉 Performance of Enhanced GA (10 × 10 problem)

Length of individual	Enhanced GA	Best Performance Before
5	**32**	**32** (GS, BS, CS)
10	**70**	**70** (BS, CS)
15	**105.2**	104.3 (GA_r1)
20	**137.4**	135.8 (GA_r1)
25	**165.6**	165.3 (GA_r1)
30	**196**	194 (CS)
35	**232.5**	230 (CS)
40	**260.4**	259 (CS)
45	**284.7**	273 (CS)
50	**308**	304.1 (GA_r1)

〈Table 9〉 Performance of Enhanced GA (20 × 20 problem)

Length of individual	Enhanced GA	Best Performance Before
5	**40**	**40** (GA_r1, GS, CS)
10	**76**	**76** (GS, BS, CS)
15	**111**	**111** (GS, BS, CS)
20	**140**	**140** (GS, BS, CS)
25	**179.3**	177 (CS)
30	**217.5**	215 (CS)
35	**253.8**	250 (CS)
40	**288.3**	283 (CS)
45	307.3	**309** (CS)
50	**241.9**	338 (CS)

〈Table 10〉 Performance of Enhanced GA (30 × 30 problem)

Length of individual	Enhanced GA	Best Performance Before
5	**39**	**39** (CS)
10	**70**	**70** (CS)
15	**104**	**104** (BS, CS)
20	**145.7**	141 (BS, CS)
25	**171.2**	168.9 (GA_r1)
30	**206.3**	204 (BS, CS)
35	**234.5**	236 (CS)
40	**270.2**	269 (CS)
45	304.7	**306** (CS)
50	**342.5**	340 (CS)

5. 결론

본 연구에서는 숫자 모으기 미로 게임에서 제한된 이동 횟수 내에 최대 점수를 획득하는 최적 경로를 탐색하기 위해 유전 알고리즘을 적용하였다. 게임판의 경계에서 이동 불가능한 방향이 발생하는 문제를 해결하기 위해 3가지 복구 기법을 비교 실험하였고, 그 결과 가장 자연스럽게 유전적 구조를 유지할 수 있는 r1 방식(가장 가까운 가능한 방향으로 보정)이 우수함을 확인하였다.

다음으로 단순 GA 알고리즘을 무작위 행동 선택, 탐욕 선택, 빔 탐색, 초쿠다이 탐색의 기존 탐색 기반 알고리즘과 비교 분석하였다. 그 결과, 단순 GA는 무작위 행동 선택 및 탐욕 탐색 대비 우수한 성능을 보였으나, 깊이 기반 탐색 능력이 강한 빔 탐색 및 초쿠다이 탐색에는 성능이 미치지 못함을 확인하였다.

이 한계를 극복하기 위해 숫자 모으기 미로 게임의 구조적 특성을 반영한 4가지 강화 기법을 도입하여 강화 GA를 설계하였다. 강화 GA를 실험한 결과, 기존 최고 성능을 능가하는 결과를 보여 주었다. 특히 개체 길이가 커질수록 (즉, 탐색해야 할 경로 공간이 커질수록) 강화 GA는 상대적으로 더 좋은 성능 향상을 보였으며, 이는 GA가 가지는 광역 탐색 능력이 더욱 복잡한 탐색 공간에서 강점을 발휘함을 의미한다. 일부 특정 개체 길이의 경우 초쿠다이 탐색이 우세했지만, 전반적으로 강화 GA는 기존 방법 대비 보다 안정적이며 높은 평균 성능을 달성하였다.

본 연구의 결과는 다음과 같은 시사점을 가진다.

첫째, 숫자 모으기 미로 게임처럼 경로 의사결정의 연속성이 강한 환경에서도 GA가 기존 알고리즘에 필적하거나 이를 능가하는 성능을 낼 수 있음을 실험적으로 확인하였다.

둘째, GA 성능은 초기화·교차·돌연변이·선택·평가 함수 등 구성 요소의 세밀한 설계에 크게 좌우되며, 본 연구에서 제안한 강화 기법들이 실제로 성능 개선에 기여함을 입증하였다.

셋째, 깊이 기반 검색(Beam/Chokudai)과 진화 기반 탐색(GA)의 특성이 서로 다르므로, 향후 두 접근 방식의 장점을 결합한 알고리즘에 대한 가능성을 제시한다.

종합적으로, 본 연구는 숫자 모으기 미로 게임에서 유전 알고리즘이 기존 탐색 알고리즘의 효과적인 대안이 될 수 있음을 보여주었다. 추후, 제안한 강화 GA가 다른 종류의 퍼즐 게임이나 일반 최적화 문제에 잘 확장되는지 확인하고, 사물인터넷 분야에 활용하기 위한 방안을 모색할 예정이다.

REFERENCES

[1] Aoki Eita, "*Introduction to Search Algorithms for Game AI*," Hanbit Media Publishing, 2024.

[2] T.Pasquier and Sobol, "How to Tailor Educational Maze Games: The Student's Preferences," *Sustainability*, pp.1-26, 2022.

[3] H.Zhang, J.Wang, W.Zhang, and Y.Wen, Y.Yu, and W.Li, "Learning to Design Games: Strategic Environments in Reinforcement Learning," *IJCAI 2018*, pp.3068-3074, 2018.

[4] S.Earle, J.Snider, M.C.Fontaine, S.Nikolaidis, and J.Togelius, "Illuminating Diverse Neural Cellular Automata for Level Generation," *arXiv*, 2021.

[5] M.G.Bellemare, S.Srinivassan, G.Ostrovski, T.Scharul, D. Saxton, and R.Munos, "Unifying Count-Based Exploration and Intrinsic Motivation," *Part of Advances Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pp.1-26, 2016.

[6] J.Schulman, F.Wolski, P.Dhariwal, A.Radford, and O.Klimov, "Proximal Policy Optimization Algorithms," *arXiv (CoRR)*, pp.1-12, 2017.

[7] N.Shaker, J.Togelius, and M.J.Nelson, "*Procedural Content Generation in Games*," Springer, 2016

[8] J.Togelius, G.N.Yannakakis, K.O.Stanley, and C.Browne, "Search-Based Procedural Content Generation," *Lecture Notes in Computer Science (LNCS)*, Vol.6024, pp.141-150, 2010.

[9] T.Pasquier and Sobol, "Genetic Algorithm Optimization in Maze Solving Problem," *Intelligent Information Management*, Vol.10, No.1, pp.1-15, 2018.

[10] A.Jonasson and S.Westerlind, "*Genetic algorithms in mazes*," Technical Report, KTH, 2016.

[11] K.Suzue, "*Adaptive NPC Behavior in Maze-Chase Game Using Genetic Algorithm*," Senior Independent Study Theses, The College of Wooster, 2022.

[12] J.Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[13] D.E.Goldberg, *Geneitc Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 2008.

[14] D.Whitley, "A Genetic Algorithm Tutorial," *Statics and Computing*, Vol.4, pp.65-85, 1994.

[15] D.A.Popescu, "An Enhanced Genetic Algorithm for Optimized Educational Assessment Test Generation Through Population Variation," *Big Data and Cognitive Computing*, Vol.9, No.4, 98 pp.1-20, 2025.

이 상 욱(Sangwook Lee) [정회원]

■ 2007년 8월 : 광주과학기술원 정보기전공학부 (공학박사)
■ 2007년 9월 ~ 2008년 9월 : 조지아공대 전산학과 박사후연구원
■ 2008년 11월 ~ 2009년 2월 : 삼성전자 통신연구소 책임연구원
■ 2009년 3월 ~ 현재 : 목원대학교 게임소프트웨어공학과 교수

〈관심분야〉
인공지능, 알고리즘, 사물인터넷, 정보통신, 게임