유전 알고리즘을 이용한 큐브 해결방법 탐구

충남중학교
30126 장현영

I. 탐구 주제

유전 알고리즘을 이용한 2x2x2 큐브의 해결방법 탐구

II. 탐구 동기 및 목적

1. 탐구 동기

과학 수업 중에 배운 염색체와 유전자에 대한 자료를 검색하던 중 유전 알고리즘이라는 탐색 알고리즘을 알게 되었다. 유전 알고리즘을 적용해 볼 만한 사례를 찾던 중 큐브가 눈에 들어와 큐브를 유전 알고리즘을 통해 풀면 어떨까라는 생각을 하게 되었다.

2. 탐구 목적

(1). 유전알고리즘을 이용해 임의의 섞인 큐브를 맞추는 해법을 도출한다.

(2). 유전알고리즘의 장단점을 파악해 더 발전시킬수 있는 점을 찾는다.

III. 유전 알고리즘의 이론과 자료 조사

1. 유전 알고리즘

유전 알고리즘은 존 홀랜드(John Holland)에 의해 개발된 기법으로, 최적화 문제를 해결하는 기법의 하나이다. 생물의 진화를 모방한 진화 연산의 대표적인 기법으로, 실제 생물의 진화 과정에서 많은 부분을 차용하였으며, 변이, 교배 연산 등이 존재한다.

일반적으로 유전 알고리즘에 대해 알고리즘이라는 표현을 이용하지만, 유전 알고리즘은 특정한 문제를 풀기 위한 알고리즘이라기 보다는 최적화 문제를 풀기 위한 방법론에 가깝다. 즉, 모든 문제에 적용 가능한 하나의 알고리즘이나 소스 코드가 있는 것이 아니기에 유전 알고리즘의 원리를 이해하고, 이를 자신이 원하는 문제에 적용할 수 있도록 하는 것이 중요하다. 유전알고리즘을 정의하는데에는 아래와 같은 개념들을 정의한다.

- **염색체 (chromosome):** 생물학적으로는 유전 물질을 담고 있는 하나의 집합을 의미하며, 유전 알고리즘에서는 하나의 해(solution)를 표현한다. 어떠한 문제의 해를 염색체로 만들어낸다.
- **유전자 (gene):** 염색체를 구성하는 요소로서, 하나의 유전 정보를 나타낸다. 어떠한 염색체가 [A B C]라면, 이 염색체에는 각각 A,B,C의 값을 갖는 3개의 유전자가 존재한다.

- **자손 (offspring):** 특정 세대 t에 존재했던 염색체들로부터 생성된 염색체를 t에 존재했던 염색체들의 자손이라고 한다. 자손은 이전 세대와 비슷한 유전 정보를 갖는다.
- **적합도 (fitness):** 어떠한 염색체가 갖고 있는 고윳값으로서, 해당 문제에 대해 염색체가 표현하는 해가 얼마나 적합한지를 나타낸다.

2. 알고리즘 구조

유전 알고리즘은 t에 존재하는 염색체들의 집합으로부터 적합도가 가장 좋은 염색체를 선택하고, 선택된 해의 방향으로 검색을 반복하면서 최적해를 찾아가는 구조로 동작한다. 유전 알고리즘의 동작을 단계별로 표현하면,

1) 초기 염색체의 집합 생성
2) 초기 염색체들에 대한 적합도 계산
3) 현재 염색체들로부터 자손을 생성
4) 생성된 자손들의 적합도 계산
5) 종료 조건 판별
6-1) 종료 조건이 거짓인 경우, (3)으로 이동
6-2) 종료 조건이 참인 경우, 알고리즘 종료

유전 알고리즘의 구조는 매우 간단하지만 (3)의 자손들을 생성하는 단계에서 여러 기법이 사용된다.

3. 자손 생성 연산

유전 알고리즘을 특정한 문제에 적용하기 위해서는 5개의 아래와 같은 연산을 정의해야한다.

- 초기 염색체 생성 연산
- 적합도 계산 연산
- 적합도를 기준으로 염색체를 선택하는 연산
- 선택된 염색체들로부터 자손을 생성하는 연산
- 돌연변이 연산

3.1 초기 염색체 생성 연산

초기 염색체 생성에서는 이전 염색체로 만들어낼 수는 없으므로 아무 규칙도 없이 임의의 값으로 염색체를 생성한다. 만약 초기 염색체를 정답에 더 가깝도록 만들 수 있다면 최적의 해를 구하는데 시간이 줄어들 수도 있다.

3.2 적합도 계산 연산

염색체에 표현된 정보를 기반으로 적합도를 계산하는 연산이다. 이 연산은 해결하고자 하는 문제에 매우 종속적이므로 다양한 연산 방법이 존재한다.

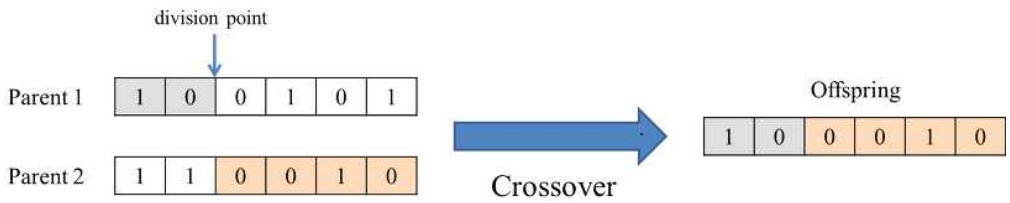
3.3 적합도를 기준으로 염색체를 선택하는 연산

한 세대에서 다음 세대로 전해지는 해의 후보가 되는 해들을 선택한다. 선택 방법에는 균등비례 룰렛 휠 선택, 토너먼트 선택, 순위 기반 선택등이 있다. 이번 탐구에서는 첫 번째 시도에서 룰렛 휠 선택을, 두 번째 시도에서 순위기반 선택을 했다.

해의 선택은 유전 알고리즘의 성능에 큰 영향을 미친다. 어떤 방법을 쓰느냐에 따라 최적해로 다가가는 속도가 더디게 되거나, 아니면 지역 최적해에 빠지는 경우가 생길 수도 있기 때문이다. 또는 우수한 해가 보유한 나쁜 인자가 전체 인구에 퍼지거나, 반대로 나쁜 해가 보유한 우수한 인자가 영구히 사장될 수도 있다. 따라서 어떤 해를 선택하는지는 유전 알고리즘의 성능을 위해서 중요한 요소라고 할 수 있다.

3.4 선택된 염색체들로부터 자손을 생성하는 연산

앞에서 선택한 부모 염색체들로부터 자손 염색체를 생성한다. 하나의 염색체로부터 임의의 유전자를 추가하는 방법, 두 개의 염색체를 교차(crossover)하여 만드는 방법이 있다. 이번 탐구에서는 첫 번째 시도에서 교차를, 두 번째 시도에서 유전자를 추가하는 방법을 택했다.



Crossover 연산의 동작

3.5 돌연변이 연산

만약, 앞에서 설명한 연산을 이용해서 검색을 진행하면, 유전 알고리즘은 그것이 정답에 멀든 가깝든 특정한 조건에 가장 가까운 지역 최적점에 도달하게 될 것이다. 유전 알고리즘에서는 정답에 먼 지역 최적점에 빠지는 문제를 해결하기 위해 새롭게 생성된 염색체에 확률적으로 돌연변이가 발생하도록 한다.

IV. 탐구 과정

1. 첫 번째 방법

첫 번째 방법에서의 알고리즘 구조는 이러하다.

1. 큐브 회전수를 20으로 설정
2. 하나의 회전을 유전자라 하여 20개의 염색체를 만들어냄
3. 각 염색체의 평가점수(적합도)를 부여해 crossover연산으로 자손 염색체 생성

먼저 임의의 섞여있는 큐브를 지정하기위해 온라인 큐브 솔빙 시간 측정 프로그램에서 얻은 임의의 scramble을 사용했다.
F2 U' R F U' F' U' R U2

이후 염색체의 평가점수를 부여하기 위해 유전자의 평가함수를 제작했다.

$$g(C_{ijn}) = \begin{cases} 10 & if\ s = 0 \\ 6 & if\ s = 2 \\ 4 & if\ s = 3 \\ 2 & if\ s = 4 \end{cases} \qquad f(Ch_{ij}) = (\frac{10000}{\prod_{n=1}^{6} g(C_{ijn})})^2$$

s = 한 면의 붙어있는 색의 개수

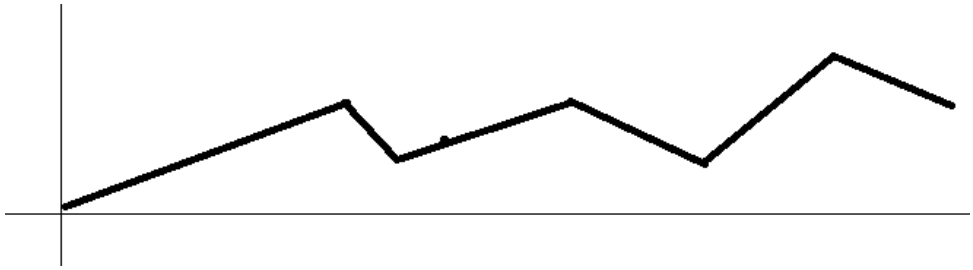
위 식에서 Ch는 유전자이며 Ch_i는 임의의 i번째 염색체를, Ch_{ij}는 임의의 i번째 염색체의 j번째 유전자를 나타낸다. 또한 C_{ijn}은 i번째 염색체의 j번째 유전자를 실행했을 때 큐브의 n번째 면을 나타낸다. 평가점수가 크면 더 좋아지도록 위 식과 같이 설정했다. 또한 점수의 차이를 더 크게 하기 위해 제곱하였다.

다음으로 염색체의 평가점수를 부여하는 평가함수를 제작했다.

$$h(Ch_i) = \frac{20\,max(Ch_i)}{j}$$

위 식에서 max함수는 염색체가 주어졌을 때 그 염색체의 위의 유전자 평가함수를 통해 얻은 점수 중 가장 높은 값을 나타낸다. 이를 분자에 놓아 점수가 크면 더 좋도록 하였으며, j는 max함수에서 얻은 최댓값이 있는 유전자의 위치로. 위치가 앞에 있을수록 후에 더 많이 회전하여 큐브를 맞출 수 있기 때문에 분모에 j를 놓아 j가 작을수록 점수가 커지게 하였다.

어느 염색체에 있는 유전자의 평가점수를 그래프화 하면

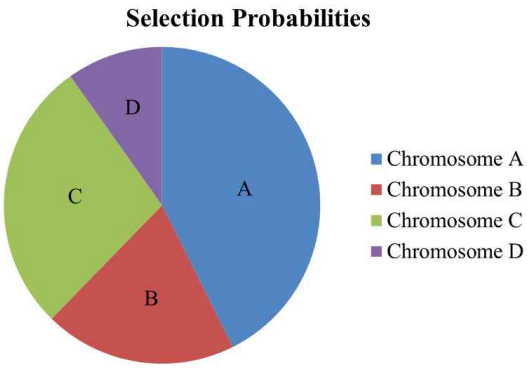


위와 같이 나올 것이라 예상하여 crossover 연산을 정의했다.

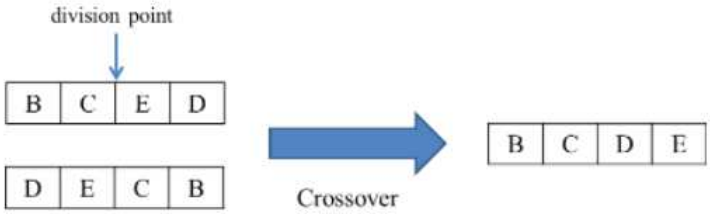
연산 방법 :

$$P(Ch_j) = \frac{f(Ch_j)}{\sum_{i}^{N} f(Ch_i)}$$

f는 염색체의 적합도를 구하는 함수



위와 같이 룰렛 휠 선택 방식으로 확률적으로 염색체를 두 개 선택하여 하나는 한 염색체의 처음부터 가장 높은 점수가 있는 n번째 유전자까지, 다른 하나는 n+1번째부터 끝까지의 유전자를 교차시킨다.



두 염색체로부터 자손 생성

생성된 염색체를 이용해 다음 세대의 연산을 반복하는 구조로 설계했다.
또한 1/1000의 확률로 유전자중 하나를 변경하는 연산도 수행했다.

2. 문제점

실행 결과 가장 큰 문제점이었던 것은 위의 그래프처럼 유전자 적합도가 나오지 않는 것이었다. 큐브를 랜덤하게 회전시키면 오히려 더 섞이는 꼴이 되어 몇 번의 시도에도 계속 모든 유전자의 점수가 1과 4에 머물러 있었다.
또한 큐브를 맞추기에 20번의 회전은 너무나 적은 수이며 한 세대 당 개체의 수도

적은 것이 문제가 되었다.

3. 두 번째 방법

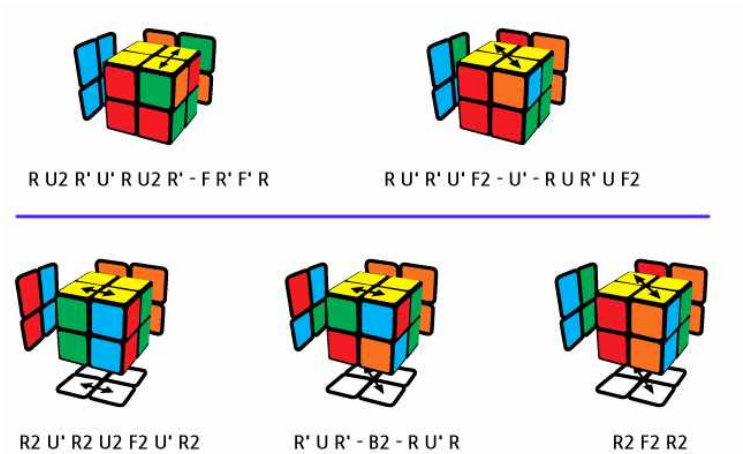
위에 서술한 문제점을 보완하여 알고리즘 구조를 세웠다.

1. 회전 횟수에 제한을 두지 않았다. 세대를 거듭할수록 회전수를 늘려 맞추게 된다.
2. 큐브에서 한번 회전하는 것은 평가점수에 아주 큰 영향을 미치므로(2x2x2큐브에서는 한번 회전할 때 큐브의 50%가 바뀐다) 회전도 하면서 큐브의 회전공식을 차용하기로 하였다.
3. 붙어있는 색의 수를 세는 게 아닌. 잘못 붙어있는 색의 수를 세어 0에 가까워지도록 하였다. 이 수를 fitness 점수로 부르기로 한다.
4. 각 세대의 개체 수는 300, 세대는 100세대까지 계산했다. 그래도 solution이 나오지 않으면 초기 염색체를 다시 만들어 반복했다.

실행 순서

1) 먼저 1세대의 초기 염색체는 임의로 두 개의 유전자를 부여하였다. 300개의 큐브를 랜덤으로 두 번 회전시켜 fitness점수가 적은 순으로 정렬한다.

2) 적은 순으로 정렬했을 때 1~50번째까지를 부모 염색체로 설정하여 유전자의 변형 없이 다음 세대에 물려주고, 다음세대의 나머지 250개의 염색체는 위의 선택된 50개의 염색체중 랜덤으로 선택하여 복사한 후 임의로 회전수를 늘린다. 이때는 임의로 한번만 회전하는 게 아닌, 큐브 공식을 사용하였다. (밑의 사진 참고)



3) 이렇게 만들어진 다음세대의 염색체를 또 다시 위와 같이 정렬하고 다음세대의 염색체를 만드는 작업을 반복한다.

이전세대의 염색체를 다음 세대에 변형 없이 물려주는 것은 회전수를 최대한 줄일 수 있을 뿐만 아니라, 좋은 유전자를 계속 이어지게 하기 위함이다.

돌연변이 연산은 큐브 솔빙에 크게 영향을 주지 않는다고 판단하여 넣지 않았다.

V. 결과

제0세대22번째 개체 / 회전수 :2
Solution
14점
L D'

제1세대75번째 개체 / 회전수 :14
Solution
11점
F' U R R U' R R U U F F U' R R

제2세대180번째 개체 / 회전수 :28
Solution
11점
R' L' R U' R' U' F F U' R U R' U F F R U U R' U' R U U R' F R' F' R

제3세대69번째 개체 / 회전수 :15
Solution
10점
F' U R U' R' U' F F U' R U R' U F F

제4세대17번째 개체 / 회전수 :11
Solution
11점
L U' y R' U R' B B R U' R

제5세대8번째 개체 / 회전수 :11
Solution
11점
L U' y R' U R' B B R U' R

제6세대166번째 개체 / 회전수 :32
Solution
7점
R B' y' x R' U R' B B R U' R y' R R F F R R R U U R' U' R U U R' F R' F' R

제7세대17번째 개체 / 회전수 :24
Solution

10점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R

(생략)

제91세대109번째 개체 / 회전수 :111
Solution
6점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R U' R' U' F F U' R U R' U F F R U' R' U' F F U' R U R' U F F R U U R' U' R U U R' F R' F' R R U U R' U' R U U R' F R' F' R R U U R' U' R U U R' F R' F' R x y' R U' R' U' F F U' R U R' U F F

제92세대1번째 개체 / 회전수 :82
Solution
7점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R R U' R R U U F F U' R R R R U' R R U U F F U' R R R U' R' U' F F U' R U R' U F F x R U U R' U' R U U R' F R' F' R

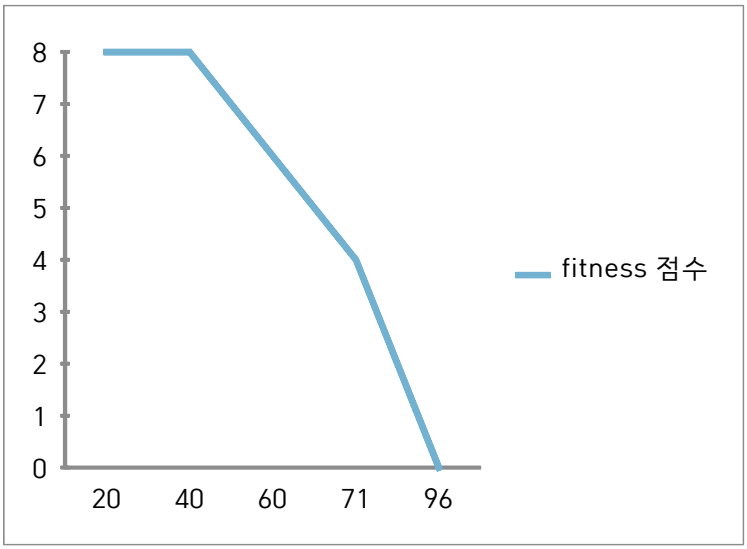
제93세대116번째 개체 / 회전수 :146
Solution
6점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R R U' R R U U F F U' R R R R U' R R U U F F U' R R R U' R' U' F F U' R U R' U F F x R U U R' U' R U U R' F R' F' R R U' R' U' F F U' R U R' U F F R U' R' U' F F U' R U R' U F F y R U' R' U' F F U' R U R' U F F R R U' R R U U F F U' R R R R U' R R U U F F U' R R

제94세대3번째 개체 / 회전수 :82
Solution
7점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R R U' R R U U F F U' R R R R U' R R U U F F U' R R R U' R' U' F F U' R U R' U F F x R U U R' U' R U U R' F R' F' R

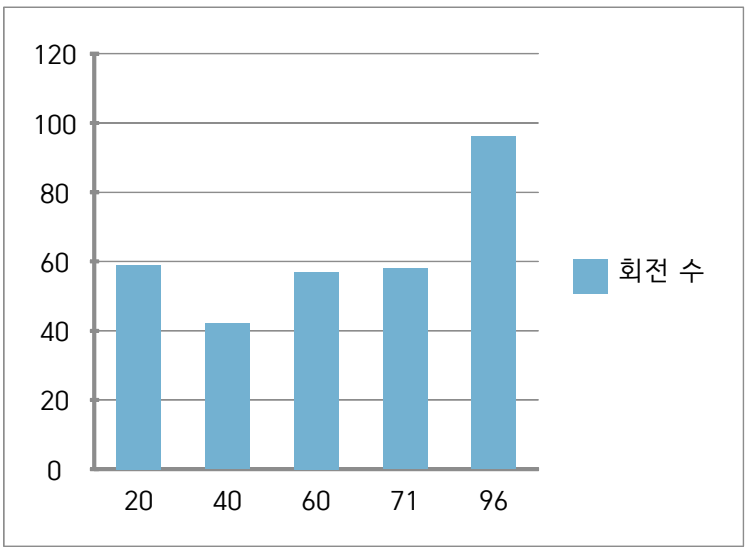
제95세대53번째 개체 / 회전수 :134
Solution
6점
L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R R U' R R U U F F U' R R R R U' R R U U F F U' R R R U' R' U' F F U' R U R' U F F x R U U R' U' R U U R' F R' F' R R U' R' U' F F U' R U R' U F F R U' R' U' F F U' R U R' U F F R U U R' U' R U U R' F R' F' R R U' R' U' F F U' R U R' U F F
Solution found
제96세대153번째 개체 / 회전수(유전자 수) :96
Solution

L U' y R' U R' B B R U' R R U U R' U' R U U R' F R' F' R z R R F F R R R R U' R R U U F F U' R R R R U' R R U U F F U' R R R U' R' U' F F U' R U R' U F F x R U U R' U' R U U R' F R' F'
R z' R U' R' U' F F U' R U R' U F F

- 세대 별 fitness점수 변화



- 세대 별 회전 수 변화



VI. 고찰

세대가 거듭하면서 fitness점수는 낮아졌다. 세대 별 회전수는 높아졌다가 낮아지기도 하다가 거의 끝에서는 급격히 오르는 추세를 보였다. 본 탐구는 c++ 언어로 큐브를 구현하고 염색체의 생성과 세대교체를 모두 자동화 하여 유전 알고리즘을 실행하는데 매우 효율적으로 진행할 수 있었다. 2x2x2 큐브보다 더 크고 경우의 수가 더 많은 3x3x3 큐브의 경우 더 많은 개체와 회전 수가 필요할 것으로 예상해볼 수 있다.

VII. 결론

탐구에서 큐브의 회전방법을 하나의 생명체에, 하나의 회전을 유전형질에 대응시켜, 생물체를 모방하여 유전이 일어날 수 있는 환경을 만들어내었다. 세대가 거듭할수록 큐브가 맞춰지고, 결국엔 큐브를 맞출 수 있음을 확인하였다. 하지만 유전 알고리즘은 가장 최적의 해를 도출 하는 것은 아니기에 최적의 해만을 고집하는 문제에서는 쓰기 어렵다는 점을 알 수 있다. 유전 알고리즘은 딥러닝이 나온 지금까지도 여전히 제 역할을 다하고 있으며 충분히 더 발전할 수 있는 가능성이 있다. 본 탐구보다 더 넓은 범위에서 더 많은 컴퓨터가 쉬지 않고 경우의 수를 탐색한다면 사람의 생각보다 더 빠르고 나은 해결방법으로 큐브를 풀 수 있는 인공지능을 만들 수 있을 것이다. 또한 자연에서 얻은 아이디어로 문제를 해결한다는 것에 큰 의의를 두고 싶다.

VIII. 탐구 활동 중 느낀 점

큐브만 구현하고 유전 알고리즘의 평가함수 식을 생각해내면서 프로그램을 쉽게 구현하며 진행할 수 있을 줄 알았다. 하지만 프로그래밍의 특성상 내가 생각하지 않은 것도 신경을 써야하며 코딩해야 하기에 유전 알고리즘 구현에만 많은 시간을 투자했다.

탐구 중 막힌 부분이 있었을 때 어떻게 해결해야 할지 막막하기도 했지만 이를 이겨내고 만족할만한 탐구 결과를 낼 수 있다는 것에 내 자신이 자랑스럽기도 하다.

프로그램 소스코드 : https://github.com/happybean4/scienceEx

- 자료 출처

https://untitledtblog.tistory.com/110 : 수식, 원형 그래프 사진

https://ko.wikipedia.org/wiki/유전_알고리즘 : 알고리즘 이론

https://blog.naver.com/vincentcube : 큐브 공식 사진