유전알고리즘 1회차 과제

2020-21659

서울대학교 컴퓨터공학부 김지연

msjiyunkim@snu.ac.kr

2022학년도 1학기

1. 과제 개괄 및 이해

- 본 과제에서는 염색체로 표현되는 문제의 최적해를 가장 가깝게 도출하는 알고리즘을 설계하였다. 이러한 목적 아래 해의 표현 방법을 결정하고, 이를 활용해 다양한 연산을 적용해보며 세대를 거듭함에 따라 알고리즘이 난수 집단으로부터 최적해에 가까워짐을 눈으로 확인하고 디자인 아이디어를 체득할 수 있었다. 보고서에서는 구현한 유전 알고리즘의 구조와 성능, 그리고 각 연산의 중요도, 연산을 할 때 섬세한 설정이 필요한 변수 요소들에 대하여 설명하고 과제를 위해 노력하면서 느꼈던 소감과 발상들을 정리한다.

2. 사용한 GA의 구조

Algorithm 1 알고리즘 설계

```
function GENETIC ALGORITHM
    random chromosome으로 이루어진 population 생성
    repeat
        for i = 1 to k do
            Population 내 모든 chromosome의 품질을 측정
            품질 상위 k개 중에서 Chromosome i1, i2 selection
            offspring(i) = uniform crossover(p1, p2)
            즉시 offspring(1), ..., offspring(k)를 population내의 품질 최하위 k개의 염색체와 replacement
        end for
    until stop condition
    가장 높은 품질의 염색체를 return
end function
```

- 본 과제에서, 유전 알고리즘 구현의 전형적인 구조를 따르되 임의로 생략 혹은 변경하여 성능의 변화를 관찰하였다. 품질 평가 방법과 이에 따른 최적해를 알고 있지만, 알고리즘은 어떤 기준으로 평가하는지, 무엇이 정답인지 알 수 없도록 설계하였다. 오직 평가 함수만이 정답에 관련한 정보를 가지고, 피드백에 의존하여 exploration과 exploitation의 균형을 추구하는 방식으로 품질을 향상시키는 유전 알고리즘의 규칙을 준수하였다.

- 다만, 전형적인 구조와 차별화되는 부분은 크게 두 가지를 들 수 있다. 첫째, mutation의 성능을 실험하고 과제에서 제거하였다. 둘째, selection-crossover-mutation을 포함하는 loop에 replacement 과정을 병행하여 메모리 및 시간 상의 구현 이점을 가져가고자 하였

다. 이러한 변형을 통하여 보다 적극적으로 유전 알고리즘에 대하여 탐구하고 실험해 볼 수 있었다.

3. 해의 표현 (Chromosome design)

- 과제에서 제공하는 해가 0과 1로 이루어진 Binary 꼴의 수들로 디자인되어 있다. 이는 전형적인 유전 알고리즘의 이진수 표현으로, 각각의 유전자가 두 가지 분류 중 어느 갈래에 속하는지 가장 간단하고 효율적으로 보여준다. 이진수는 코딩에 있어 간단하고 기초적인 표현이지만, 다양하고 유연한 스키마 처리를 돕는다.
- 이진수의 계산적인 이점을 최대화하기 위하여, 본 과제의 구현에서는 bitset (비트셋) 자료형을 채택하여 메모리를 절약하고 계산 속도를 향상시키고자 하였다. Bitset 자료형은 고정 크기의 n개 bit를 가지는 시퀀스로, 논리연산과 bit flip 등을 지원하여 과제에 적합하였다.

	염색체 (chromosome)	해집단 (population)	유전자 (gene)
표현	Bitset	Bitset vector	Bit

4. 연산자에 대한 설명 (Selection, crossover, mutation, replacement 등)

- **Selection**

Algorithm 2 단순화된 순위 기반 선택

```
function SELECTION(Population)
    Q = chronosome들의 품질을 저장하는 배열
    for a chronosome c in population do
        qual = 품질평가함수(c)
        Q에 qual을 저장
    end for
    p1 = Q를 바탕으로 Population 안 품질 상위 q개의 해 중에서 무작위로 선택
    p2 = p1과 같은 품질 상위 q개의 해 중에서 무작위로 선택
end function
```

- 순위 기반 선택 : 순위 기반 선택이란 해집단 내의 해들을 높은 품질부터 순위 매겨 높은 품질의 해일수록 선택 시 더 많이 고려하는 방법이다. 이를 통해 해들은 최대 적합도와 최소 적합도 사이에서 균일한 간격의 적합도 분포를 가지게 된다.
- 순위 기반 선택을 선정한 이유 : 본 과제에서 순위 기반 선택을 채택한 가장 큰 이유는 대표적인 선택 방법인 품질 비례 룰렛휠 선택에 비해 해들이 적합도를 고르게 가지게 되고, 보다 안정적이고 양질의 해 선택이 가능할 것이라고 추론하였기 때문이다. 또한, sorting과 slicing 등의 기능을 통해 직관적으로 구현할 수

있었다.

- 순위 기반 선택을 단순화한 이유 : 본래 순위 기반 선택은 적합도를 배정하여 선택의 당위를 부여하는데, 본 과제에서는 적합도를 구하는 식을 사용하지 않고 품질 함수를 통해 구해진 값을 바탕으로 우선순위를 할당하였다. 결과적으로 구현하고자 하는 바가 유사한데 비해 보다 빠르고 쉽게 선택할 수 있는 방법이라고 생각하였다.

- 변형 아이디어 : 순위 기반 선택과 더불어, 몇 가지 최적화 및 파라미터 조정 방법을 통하여 보다 좋은 선택을 할 수 있을 것이다. 예를 들어, 난수 생성으로 뽑은 두 개의 chromosome이 우연의 일치로 같은 chromosome일 경우, 불필요한 계산을 하거나 확률의 장난으로 다음 세대에 저품질의 해가 너무 쉽게 진출하는 일을 지양하기 위해 같은 chromosome끼리 선택하지 않는 보조 메소드를 생성할 수 있다. 또한, 고품질 해들의 풀(pool)을 몇 개 정도로 잡는지에 따라 더 빨리 좋은 값으로 수렴할 것인지, 좋지 않은 값에서 수렴할 것인지 실험할 수 있다. 무조건 고품질의 해들만 고른다고 하여 그 교배 결과로 더 좋은 품질의 해가 나오는 것은 아니기 때문이다. 다양성을 함께 고려하는 것이 보다 빠른 최적해 수렴으로 이어질 수 있다.

- **Crossover**

Algorithm 3 균등 교차

```
1: function SELECTION(c1, c2)          ▷ c1, c2 - Selection의 결과인 parent chronosomes
2:     n = 새로운 chromosome
3:     for i = 0 to (chromosome 길이-1) do
4:         난수 발생
5:         if 난수가 홀수 then
6:             c1에서 i번째 자리 수를 가져와 n에 이식
7:         else
8:             c2에서 i번째 자리 수를 가져와 n에 이식
9:         end if
10:    end for
11: end function
```

- 균등 교차 : 일반적인 균등 교차는 일정 수 혹은 확률을 임의로 설정하고 난수를 발생시켜 난수가 그 수를 넘는지 여부에 따라 부모 chromosomes 중 어느 쪽의 유전자를 복사한다. 세대가 진행될수록 해집단이 수렴하여 결국 대부분의 해들이 같거나 유사하게 만들어진다는 특징이 있다.

- 균등 교차를 선정한 이유 : 균등 교차 역시 일점 교차나 다점 교차에 비해 다양하고 유연한 스키마를 만들 수 있다. 또한, 일점 교차 혹은 다점 교차의 구현 시에 마스크 등을 이용해야 하지만, 균등 교차는 조금 시간이 오래 걸리더라도 간단한 구현과 강력한 수렴 성능을 보여주었으므로 균등 교차를 채택하였다.

■ 변형 아이디어 : 균등교차를 변형시켜 같은 원리이되 부모 노드의 품질에 따라 자식에게 유전자를 물려줄 확률을 조정하는 하이브리드 메소드가 가능할 것이다.

- **Mutation**

Algorithm 4 전형적 변이

```
1: function MUTATION
2:     c = Chromosome의 크기
3:     h = 임의로 지정한 일정 확률
4:     for i = 1 to c do
5:         r = 난수 확률
6:         if r < h then
7:             Chromosome 내의 i번째 자리 유전자의 값을 다른 값으로 수정
8:         end if
9:     end for
10: end function
```

■ 전형적 변이 : Chromosome 내의 각각 유전자 자리마다 그 자리의 유전자를 다른 유전자로 바꿀지에 대해 랜덤하게 결정한다.

■ 전형적 변이를 선정한 이유 : 부모 세대에서 가지고 있는 특징들 만을 이용하여 자식 세대를 만들 경우 탐색 공간이 제한되고 local minima에서 수렴할 수 있다.

■ 삭제 이유 : 변이는 이미 잘 만들어진 스키마에 구멍을 내는 요인이므로 실험 결과 해의 품질을 거의 항상 떨어뜨리는 것을 확인하였다. 또한 수렴 성향이 강한 균등 교차를 이전 단계에서 채택하였는데, 동시에 강한 변이를 넣었더니 품질 향상이 잘 이루어지지 않았다.

- **Replacement**

Algorithm 5 강한 안정 상태의 대치

```
1: function REPLACE(Population, Offspring)        ▷ Population - 현재 해 집단, Offspring - 자식 세대
2:     P(population) = Population 내 모든 chromosomes의 품질을 평가한 set
3:     P(population)에서 가장 품질이 낮은 chromosome과 새로 만든 Offspring을 대치
4: end function
```

■ 강한 안정 상태의 대치법 : 교배를 통한 해가 생기면 그 해를 즉시 population 내의 해 하나와 대치한다. Population에서 가장 품질이 낮은, 의미가 없는 해부터 대치하여 population 내에서 가장 품질이 좋은 해는 대치되어 없어지지 않도록 고려하였다.

■ 이와 같은 대치를 고른 이유 : selection, crossover와 동시에 replace해줄 경우 편리하고 간단하므로 선택하였다. 그러나 이 경우 수렴이 빠르게 진행되어 local minima에 빠지는 어려움이 있었다.

5. Result table

- Setting : 30초 미만의 유전 알고리즘 실행, Initial population , top k개 선정 메커니즘은 동일하다.

	n = 8	n = 32	n = 80
Best performance	47	198	447
Avg. performance	41.5	154.3	383.7
Standard deviation	4.23	10.23	17.42

6. 세대 진행에 따른 Population 분석

- 본 단원에서는 4의 sample runs중 가장 큰 샘플 케이스 (n = 80) 를 선택하여 세대 진행에 따른 population을 분석한다.

	100번째 세대	300번째 세대	500번째 세대
Best performance	397	410	442
Avg. performance	392	401	420
Standard deviation	10.34	8.24	5.15

- 세대가 진행됨에 따라 해의 평균 품질과 최고 품질은 세대가 진행될수록 수렴하는 양상을 보인다.

7. 관찰 아이디어 제언

- 5점 스키마와 비교하였을 때, 8점 스키마는 그보다 중요하고 정보량이 크다. 그러므로, 점점 좋은 chromosome이 만들어지는 것을 관찰하기 위하여 세대가 거듭할수록 더 빨리, 더 자주 나타나는 스키마가 5점인지 8점인지 계산해보면 뚜렷하게 드러날 것이라 추측하였다.
- 관찰 내용
  - ■ 스키마의 밸류들이 8 혹은 5로만 이루어져 있다는 점에서 착안하여, GA를 실행하고 최종적으로 도출되는 score들을 8과 5의 곱으로 factorizing하고 더하면, 총 8의 등장 횟수가 5의 등장 횟수보다 월등히 많은 것을 확인할 수 있다.

8. Discussion

- 느낀 점 : 처음 품질을 검사하였을 때 거의 모든 chromosome에서 검사 값이 0인 것을 발견하고, 품질의 기준조차 모델이 모르도록 해야 하는 상황에서 이 부족한 정보들 속 어떻게 해의 품질이 향상될지 매우 의심스러웠다. 그러나 단순하지만 확실하게

점진적으로 품질을 발전시켜 나가는 모습이 경이롭기도 하고 매우 흥미로웠다.

- 잘 안되는 점 : 앞서 느낀 점과 같이, 희박한 정보 속에서 피드백을 바탕으로 발전하는 알고리즘이다보니, 초반에 생성되는 random population을 비롯해 parameter의 영향을 매우 쉽게 타는 것을 느꼈다. 또한 연산의 개수가 많아 주어진 시간 내에 task가 완료되도록 설계할 수 있을지 매우 노파심이 들었다.
- 의외의 현상 : 구현과 실험 중간에 알고리즘 아주 초반부터 품질 값이 고정되어 알고리즘이 끝날 때까지 한 값으로 유지되는 현상이 있었다. 이러한 현상은 local minima에서 일찍이 수렴하였기 때문에 나타났다는 것을 확인하였다.
- 예상대로 된 점 : 유전 알고리즘을 처음으로 구현해보는 경험이라 예상대로 된 경우는 많지 않았지만, replacement를 selection/crossover와 같은 loop 안에서 돌렸을 때보다 공간 및 시간 효율적으로 좋을 것이라고 추측했고, 실제로 성능이 떨어지지 않았다.

9. Conclusion

- 본 과제에서는 순수 GA만 사용하여 royal-road function을 통해 그 한계를 경험하는 전형적인 프로세스를 구현하였다. 이로써 유전 알고리즘의 품질 향상 과정을 눈으로 관찰하고 성능에 중요한 영향을 미치는 요인들로 selection/crossover/mutation/replacement 전략 채택, population 수, chronosome의 길이, generation의 수 등 다양한 분야의 많은 선택들이 있다는 것을 알게 되었다.

10.Reference

- 문병로. 쉽게 배우는 유전 알고리즘. 한빛 미디어, 2008.