

유전 알고리즘 Project 3 Report

Maxcut , hybrid GA

2020-22384 김정현, 2020-21659 김지연

1. 서론

이 보고서는 서울대학교 컴퓨터공학부 2022년 1학기 유전 알고리즘 과목의 세 번째 프로젝트의 결과물에 대한 보고서이다. 이전 과제에서 graph maximum cut 문제에 대하여 순수한 유전 알고리즘을 통해 제한 시간 안에 최적해를 구하고자 하였다만, 이번 과제에서는 이를 심화하여 같은 문제에 대해 하이브리드 유전 알고리즘을 적용해본다. 순수 유전 알고리즘에 비하여, 지역 최적화를 적용한 하이브리드 유전 알고리즘(미미틱 유전 알고리즘)의 경우 보다 나은 성능을 비롯하여 적은 population 수 등 여러 이점을 보였다. 본 보고서에서는 문제의 해답을 제시하고, 그 결과와 시사점을 정리한다. 기본 유전 알고리즘의 4가지 테크닉인 선택, 교차, 변이, 대치, 그리고 지역최적화 기법들을 구상하여 각각의 단계에서 어떤 방법론을 적용하였는지 깊이있게 다룬다. 추가적으로, 서로 다른 환경의 데이터셋들을 바탕으로 실험하여 모델의 범용성 및 유효성을 검증하였다. 마지막으로, 이번 과제를 통하여 느낀 점, 예상과 다른 결과를 냈던 부분, 어려웠던 부분 등을 회고하였다.

2. 전체 GA 구조

본 과제에서 사용한 알고리즘의 전체적인 구조는 아래의 Algorithm 1로 요약한다.

Algorithm 1 전체 GA 구조

```
1: function HYBRID GA OF TEAM 17
2:    $P_1$  = subpopulation 1의 chromosome 수
3:    $P_2$  = subpopulation 2의 chromosome 수
4:    $P_3$  = subpopulation 3의 chromosome 수
5:    $P_1, P_2, P_3$  크기의 random chromosomes으로 이루어진 initial subpopulations를 생성
6:   repeat
7:     for generation do
8:       for subpopulation do
9:         if subpopulation1 then
10:           subpopulation 내 모든 chromosome의 품질을 측정
11:           품질 상위  $P_1 * 3 / 5$ 개 중에서 chromosome  $i_1, i_2$  selection
12:           offspring(i) = uniform crossover( $i_1, i_2$ )
13:           즉시 offspring(i)를 population 내의 품질이 i번째로 낮은 염색체와 replacement
14:           if converged then subpopulation2 = subpopulation1 + subpopulation
15:           end if
16:         end if
17:         if subpopulation2 then
18:           subpopulation 내 모든 chromosome의 품질을 측정
19:           품질 상위  $P_2 * 3 / 5$ 개 중에서 chromosome  $i_1, i_2$  selection
20:           offspring(i) = multi-point crossover( $i_1, i_2$ )
21:           즉시 offspring(i)를 population내의 품질이 i번째로 낮은 염색체와 replacement
22:           if converged then subpopulation3 = subpopulation2 + subpopulation
23:           end if
24:         end if
25:         if subpopulation3 then
26:           subpopulation 내 모든 chromosome의 품질을 측정
27:           품질 상위  $P_3 * 3 / 5$ 개 중에서 chromosome  $i_1, i_2$  selection
28:           offspring(i) = multi-point crossover( $i_1, i_2$ )
29:           즉시 offspring(i)를 population내의 품질이 i번째로 낮은 염색체와 replacement
30:           if converged then subpopulation1 = subpopulation3 + subpopulation
31:           end if
32:         end if
33:       end for
34:     end for
35:   until stopping condition
36:   가장 높은 품질의 chromosome에 MAX-LG 기법 적용
37:   MAX-LG를 적용한 chromosome을 반환
38: end function
```

3. 지역 최적화 기법 / 연산자 고안

3.1. 지역최적화 기법

본 과제에서, 크게 2가지 지역 최적화 기법이 적용되었다.

3.1.1. Island method (섬식 방법)

3.1.1.1. Intuition

Island method는 전체 population을 여러 개로 쪼개어, 각 subpopulation이 독립적으로 유전 알고리즘을 시행하다가 특정 조건 아래 서로 발전시킨 내용을 교류하는 방법들의 통칭이다. 이와 같은 방법을 통하여, 각 subpopulation에서 local optimum으로 수렴했을 때 이를 벗어나기 위한 기회를 제공한다. 반드시 global optimum으로 간다는 보장은 없지만, 한나의 population을 운용할 때에 비하여 빠르게 local optimum으로 가는 것을 방지한다.

3.1.1.2. 알고리즘

본 과제에서 섬식 방법을 적용할 때, 방법의 이점을 최대한 활용하기 위하여 population과 submodel architecture를 다양하게 설정하여 폭넓은 hyperparameter search를 진행하였다. 그 결과, 지난 과제(pj2)와 동일하게 세 subpopulation 모두에 대하여 non-mutation 기조를 유지하되, 각각에게 다른 crossover 법칙을 적용하고 먼저 수렴하는 subpopulation을 다른 subpopulation으로 병합하도록 하였다. 예를 들어, subpopulation0이 수렴하면 subpopulation1에 합쳐지고, subpopulation1이 수렴하면 subpopulation2에 합쳐지고, subpopulation2가 수렴하면 subpopulation0에 합쳐지는 방식으로 각각 한 번씩 수렴하여 서로가 서로를 삼각형 꼴로 골고루 참조할 수 있도록 설계하였다.

3.1.2. MAX-LG method

3.1.2.1. Intuition

MAX-LG(Lock Gain) 기반의 알고리즘은 maxcut problem을 해결하기 위한 컷사이즈(cut size) 증가를 lock gain으로 정의한 뒤, 이 성질에 초점을 맞추어 지역 최적화를 꾀하는 기법이다. Locked gains 정보를 저장하여 chromosome 내에서 영향력이 큰 gene(bit)들을 하나씩 수정하여 보다 높은 품질의 해를 산출한다.

3.1.2.2. 알고리즘

Algorithm 3: MAX-LG

Input: a chromosome x
Output: an optimized chromosome opt_x

```
1: Initialize:
   int array scores
2:  $prev\_score \leftarrow compute\_score(x)$ 
3: while true do
4:   for unlocked nodes  $i$  of  $x$  do
5:     flip ( $x[i]$ )
6:     scores.append( $compute\_score(x[i])$ )
7:     flip ( $x[i]$ )
8:   end for
9:    $max\_index \leftarrow find\_max\_index(scores)$ 
10:   $new\_score \leftarrow x[max\_index]$ 
11:  lock  $x[max\_index]$ 
12:  if  $new\_score < prev\_score$  then
13:    break
14:  else
15:     $prev\_score \leftarrow new\_score$ 
16:     $opt\_x \leftarrow x$  of  $new\_score$ 
17:  end if
18: end while
19: return  $opt\_x$ 
```

3.2. 연산자 설명

3.2.1. Top-k selection

Selection 과정에서는 population에서 두 개의 부모 유전자를 추출한다. 추출 방식은 Algorithm 2.에서 자세히 기술한다.

Algorithm 2 두 부모해 선택 (Selection)

```
1: sorted_index ← population의 index를 score를 기준으로 descending order sort
2: inner_idx1, inner_idx2를 [0, k]에서 random하게 뽑는다.
3: idx1 ← sorted_index[inner_idx1]
4: idx2 ← sorted_index[inner_idx2]
5: p1 ← population[idx1]
6: p2 ← population[idx2]
```

우리는 population으로부터 두 부모 유전자를 선정하는 방식으로 top-K random selection 방식을 사용하였다. 이 방식은 population의 모든 유전자에 대한 score를 구한 후, top-K scores를 가지는 유전자들로부터 랜덤하게 두 부모 유전자를 select하는 방식이다. Top-K random selection 방식 뿐만 아니라 전체 population에서 랜덤하게 두 부모 유전자를 선택하는 naive random selection, 한 부모는 top-K scores를 가지는 유전자들로부터, 다른 한 부모는 나머지 유전자들로부터 받아오는 hybrid 방식 역시 실험해본 결과, hybrid 방식이 타 방식들보다 weighted_500 데이터셋 기준 근소하게 높은 성능을 보였으나, top-k random selection에 비해 조금 더 분산이 컸다. Top-K random selection, hybrid 양쪽 모두 각각의 장단점이 있었다.

3.2.2. Various crossover techniques

Crossover 과정에서는 두 부모 유전자를 교배하여 자식 유전자 offspring를 생성한다. 총 두 가지 crossover 방식, uniform crossover, multi-point crossover 방식이 각각의 island에 적용된다. 이 중 uniform crossover 과정은 Algorithm 3.에서 자세히 기술한다.

Algorithm 3 교배 (cross-over function)

```
1: function CROSS_OVER(p1, p2)
2:   N ← 각 해의 길이
3:   pnew ← []
4:   for i = 1 to N do
5:     idx에 0 or 1을 랜덤하게 부여
6:     if idx==1 then
7:       pnew[i] ← p1[i]
8:     else
9:       pnew[i] ← p2[i]
10:    end if
11:   end for
12: end function
```

Uniform crossover 방식은 유전자의 각 자리별로 어떤 부모로부터 정보를 받아올 지 랜덤하게 결정하는데, 이를 위해 idx 변수에 0 또는 1을 랜덤하게 부여하여 이 변수를 기반으로 부모를 선택, 유전을 진행한다. Single point crossover 방식은 유전자의 특정한 한 위치를 기준으로 한 쪽은 첫 번째 부모로부터, 다른 쪽은 두 번째 부모로부터 정보를 받아온다. Multi-point crossover 방식은 유전자의 특정 여러 위치를 기준으로 두 부모로부터 교차적으로 정보를 받아온다.

3.2.3. No mutation

Mutation 과정은 생성된 자식 유전자에 변이를 부여한다. 이 과정은 Algorithm 4. 에서 자세히 기술한다.

Algorithm 4 변이 (mutation function)

```
1: function MUTATION( $p_{\text{new}}$ )
2:    $N \leftarrow$  각 해의 길이
3:   for  $i = 1$  to  $N$  do
4:     float flag 를  $[0, 100]$ 에서 random 하게 뽑는다.
5:     if flag  $\leq 5$  then
6:       flip  $p_{\text{new}}[i]$ 
7:     end if
8:   end for
9: end function
```

우리는 mutation 기법으로 uniform flip 방식을 사용하였다. 이 방식은 특정 확률에 따라 자식 유전자의 각 자리 별로 bit flip 을 진행하여 변이를 부여하는데, 이는 population 에 돌연변이를 탄생시키는 역할을 한다. 그러나 10 번의 실험 결과 모델의 수렴 속도가 느려지고, 평균적인 성능 또한 낮아지는 것을 확인하였다. 우리는 균등 변이에 더해 비균등 변이 방식 또한 순수 유전 모델에 적용하여 성능을 검증해보았다. 비균등 변이 방식은 균등 변이 방식의 bit flip 확률을 세대가 진행됨에 따라 점진적으로 줄여주는 방식을 사용한다. 이는 세대가 거듭될수록 품질이 좋은 자식 유전자가 탄생할 것을 가정하는데, 실험 결과 local minimum에 수렴하여 premature convergence 문제가 생기면서 자식 유전자의 품질에 악영향을 주었다. 이러한 고려를 거쳐, 성능의 향상을 위해 전략적으로 mutation 과정을 생략하여 더 좋은 성능을 얻을 수 있었다.

3.2.4. Replacement with low-score chromosomes

Replacement 과정에서는 생성된 자식 유전자를 population의 기존 유전자와 대치한다. 이 과정은 Algorithm 5. 에 기술한다.

Algorithm 5 강한 안정 상태의 대치

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

Population 내에서 가장 품질이 낮은, 의미가 없는 해부터 대치하여 가장 품질이 좋은 해는 대치되어 없어지지 않도록 고려한다. 이 때 selection, crossover 후에 생긴 offspring 를 바로 replace 할 경우 편리하고 간단한 장점이 있다.

4. 실험

4.1. 8개 모두에 대해 알고리즘이 얻는 해의 품질

샘플로 주어진 8개의 데이터셋에 대하여, 개발한 하이브리드 유전 알고리즘을 각 20번 수행하여 가장 좋은 결과, 평균 결과, 표준 편차 등을 정리하였다. 또한, 각 데이터셋의 특성에 따라 성능에 어떤 차이가 있었는지 분석하고 그 원인을 추론하였다.
(Table 1에서, $\text{Density} = 2|E|/(|V|(|V|-1))$)

Table 1. Dataset description

Data	V	E	Density	Mean deg.	Max deg.	Std. of weights
g1	946	3,240	0.0072	6	945	75.72
g2	1,000	3,000	0.0060	5	6	1.00
g3	1,000	10,000	0.0200	19	33	1.48
g4	1,852	6,480	0.0038	6	945	77.03
g5	2,000	4,000	0.0020	3	4	1.00
g6	2,000	19,990	0.0100	19	40	0
g7	2,000	11,778	0.0059	11	210	0
g8	2,744	8,232	0.0022	5	6	1

Table 2. Performance

Data	Optimal	Max	Min	Avg.	Std.	Max/Opt.	Avg./Opt.
g1	31,856	28,976	28,976	28,976	0	0.9096	0.9096
g2	894	800	774	790.5	11.4	0.8949	0.8842
g3	9,524	9,285	8,520	9,030	305.0	0.9749	0.9481
g4	65,864	57,096	57,096	57,096	0	0.8669	0.8669
g5	1,410	1,264	1,208	1,241	20.4	0.8965	0.8801
g6	13,328	12,921	12,800	12844.5	47.9	0.9695	0.9637
g7	7,687	7,547	7,286	7423.3	101.5	0.9818	0.9657
g8	2,460	2,203	2,151	2183.5	20.1	0.8956	0.8876

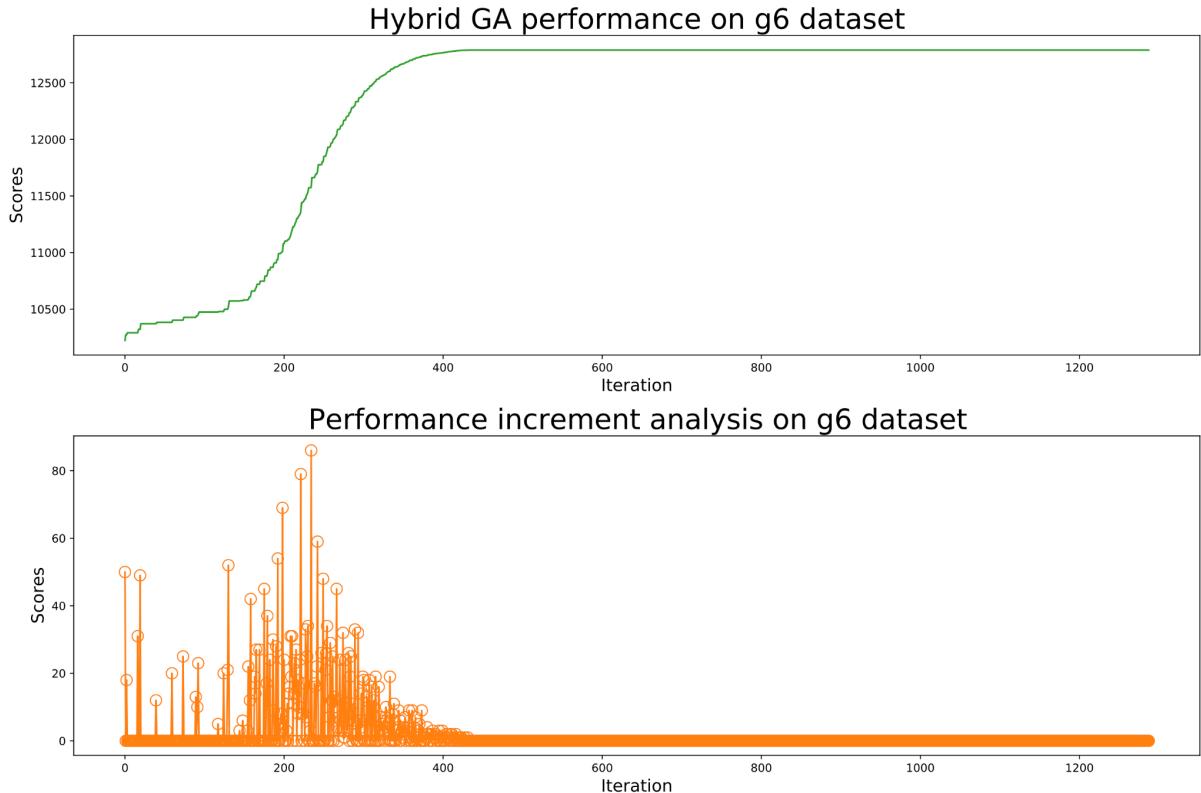
- g6 dataset의 경우, initialized score에 비해 GA가 계산해낸 final score가 그렇게 높지 않았는데, 이에 대하여 정점의 수에 비해 간선의 수가 많은 dense 한 그래프이므로 어떻게 자르든 cut에 따른 점수가 크게 달라지지 않을 것이라는 분석이 가능하다.
- Weights가 비교적 큰 g1, g4 데이터의 경우 해가 조금 변할 때 score (sum of weights of cut edges) 가 크게 변하므로, max performance의 표준 편차가 클 것이라는 분석이 가능하다.
- Max degree가 큰 g1, g4 데이터의 경우는 power law를 따르는 social network 형태의 그래프 구조일 확률이 높다. 따라서 graph cut을 최대화하려면, 이러한 influencer node 가 속한 집합은 소수 노드 집합일 확률이 높다고 예상하였다.

4.2. 가장 큰 그래프 관찰

4.2.1. 수렴성

우리는 샘플로 주어진 8개의 데이터 중 가장 큰 그래프 (g8)에 대해 1) GA 수행에 따른 population의 수렴성을 관찰하고, 2) 이를 multi-start 기법의 성능과 비교한다. Island 기법에서 우리는 3개의 island를 사용하므로 multi-start 기법에서도 3 개의 독립적인 GA를 수행한다.

GA 수행에 따른 population의 수렴성을 관찰하기 위해 우리는 세대 별 max score의 변화에 대해 세대 별 max score의 변화량을 리포트한다. 결과는 아래 그림에서 확인할 수 있다.



위 결과에서 두 번째 그림, iteration 별 max score의 증가량을 보면, 중간 일정 지점에서 크게 증가하는 것을 확인할 수 있다. 이는 Local optimization 기법을 사용하지 않은 GA 모델이 보통 균일하게 score가 증가하는 것과 비교된다.

4.2.2. 유전 알고리즘의 효과

Multi-start 기법은 population의 해를 일정 조건이 충족되었을 때 초기화해주는 알고리즘이다. 우리는 multi-start 기법과 MAX-LG 기법을 사용한 모델을 GA와 MAX-LG 기법을 사용한 모델과 비교하여, GA 과정의 중요성을 확인한다.

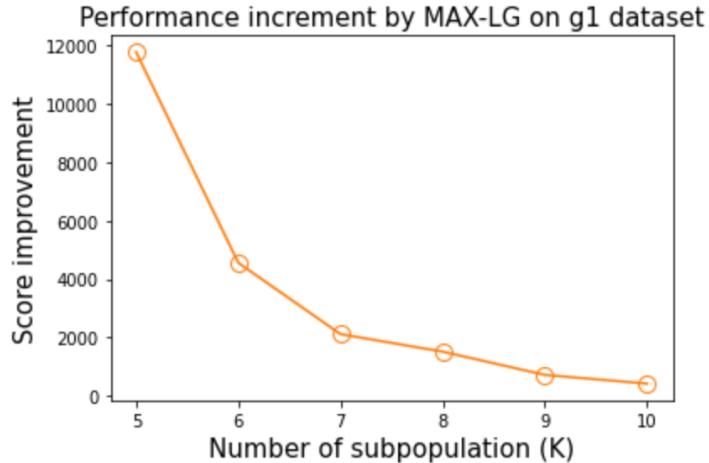
Model	Max score	Running time (sec)
Multi-start + MAX-LG	12,862	2,640
GA + MAX-LG	12,921	180

표의 결과를 보면 GA 과정 없이 multi-start 기법을 사용한 모델이 근소하게 더 낮은 것을 확인했다. 이는 우리의 모델에서는 GA의 최종 max-score-chromosome에 MAX-LG를 한 번 사용하여 해를 tuning 해주는 목적으로 local optimization 기법이 사용되었다면, multi-start + MAX-LG 모델에서는 처음부터 최적 해를 찾는 목적으로 local optimization 기법이 사용되었기 때문에 낮은 품질의 local minima에 빠진 것을 확인할 수 있다.. 또한, multi-start + MAX-LG 모델은 random initialized chromosome에 local optimization

기법이 사용되었기 때문에 수렴점을 찾기까지 매우 긴 시간이 필요한 것을 관찰할 수 있었다.

4.3. 기타 GA의 작동 메커니즘을 관찰할 수 있는 독자적 실험

우리는 population의 수와 MAX-LG 기법의 상관 관계를 알아보는 실험을 진행한다. 실험은 g1 데이터에서 진행하며, population 수를 500, 600, ..., 1000 까지 늘려가면서 MAX-LG 가 적용되었을 때 해의 품질 변화를 max score 를 통해 분석한다.



위의 그림을 통해 우리는 population number 가 커질수록 local optimization 의 영향력이 exponential 하게 적어지는 것을 확인할 수 있었다.

5. Discussion

5.1. 느낀 점

이전 과제와 유사한 설정에서 지역 최적화를 추가해보는 경험을 통해 순수 유전 알고리즘이 가지는 한계를 극복하기 위한 여러 기법을 공부해볼 수 있어 의미있었다. 더 나아가, 지역최적화를 적용하는 과정에서도 선택압을 고려한 조절이 필요함을 알게되었다. 섬식 기법을 적용하기에 앞서, 섬(subpopulation) 별로 어떤 기법을 적용해야 할지, 그리고 각 섬(subpopulation)들을 어떻게 서로 참조시킬지 많은 토론을 하였다. 토론 결과 유전알고리즘의 전체적인 구조를 유기적으로 구성하는 것이 얼마나 중요한지 깨닫게되었다.

5.2. 잘 안 되는 점

본 과제에서 가장 까다로웠던 부분은 지역최적화 기법을 추가하되, 이전 과제보다 더 커진 데이터셋을 제한된 시간과 자원을 바탕으로 효과적이고 안정적으로 처리하기 위해 연산량을 조절하는 부분이었다. 지역최적화 기법들은 대부분 구해진 해가 local optimum인지 확인해보기 위해 다른 subpopulation과 비교하거나 (섬식 방법) 각 정점의 위치를 바꾸어 시뮬레이션 해보는 (MAX-LG) 과정을 거치는데, 이 단계에서 정점의 수와 간선의 수에 비례하는 연산량의 증가가 일어날 수 있기 때문에 매 세대마다 이를 수행해보기에는 무리가 있다는 생각이 들었다. 게다가, 이번 과제의 경우 간선의 수와 정점의 수가 대폭 늘어난데다가 주어지는 샘플 데이터셋도 이전 과제에 비해 분산이 커서 이 데이터들에서 고루 안정적인 성능을 낼 수 있을만한 기법을 고안하는 데에 어려움이 있었다.

5.3. 의외의 현상

기존 과제(pj2)의 구현 내용을 바탕으로 지역최적화를 적용하였을 때, island method에서 각 subpopulation이 서로를 참고하였음에도 불구하고 성능개선이 이루어지지 않았다. 그 이유에 대하여 다음과 같은 분석이 가능하였다.

- 기존 과제에서 채택한 pureGA의 구조는 top-k selection, uniform crossover, no mutation, replacement with lowest chromosomes 이다.

- Subpopulation 별로 다른 genetic algorithm architecture를 사용하고, 각각을 독립적으로 발전시키는 단계에서 각 subpopulation의 score가 올라가고 수렴하는 속도에 차이가 생긴다.
- 그러나 subpopulation이 수렴하고 서로를 참조하는 단계에서, 이미 빠른 발전을 한 subpopulation의 해의 전체적인 품질이 그렇지 않은 subpopulation에 비해 월등히 높은 경우가 발생하고, 이 때 top-k selection 및 replacement with lowest chromosomes 단계에서 느린 발전을 한 subpopulation의 해들이 쉽게 삭제된다.
- 그러므로, 전체 해들은 concatenation을 거쳐 커진 population의 규모에 비해 해의 다양성을 보장받지 못하였다.

5.4. 예상대로 된 점

참고문헌 조사를 통해 MAX-LG를 발견하였을 때, 알고리즘을 읽으며 연산량이 다소 크지만 반드시 해의 품질을 올릴 수 밖에 없는 알고리즘이라고 판단하여 적용하였다. 예상과 같이, lock gain에 중점을 맞추어 해의 품질이 좋아지는 방향으로 chromosome을 변형하는 MAX-LG 지역최적화 방법을 알고리즘의 마지막에 반복적이지 않은 방향으로 적용하여 연산량의 무리 없이 성공적으로 해의 품질을 올릴 수 있었다.

6. 결론

본 보고서를 통해 graph maximum cut 문제를 하이브리드 유전 알고리즘을 통해 해결할 수 있음을 보이고, 그 구현 디테일을 비롯하여 해결 과정에서 새롭게 구상한 지역 최적화 메소드 등을 논의하였다. Local optimum에서 위하여, Island Method, MAX-LG 등 지역 최적화 방법들을 도입하여 실험해보았다. 그 결과 샘플로 주어진 8개의 그래프 맥스컷을 학습시켰을 때 최적해에 대비하여 전체적으로 높은 정확도를 보였으며, 그 이유에 대해 분석하였다.

7. 기여율

김정현(2020-22384) 50%, 김지연(2020-21659) 50%

8. Reference

1. 문병로, 쉽게 배우는 유전 알고리즘, 한빛 미디어, 2008.
2. 김수향 외., A Hybrid Genetic Algorithm for the MAX CUT Problem, GENETIC ALGORITHMS, 2001.