

2018 GA Project 2: MAX-CUT with 1-Bit Flip

PyeongSeok Oh
Computer Science and Engineering
Seoul National University
2017-20870
ohasno@naver.com

I. INTRODUCTION

이번 프로젝트에서는 MAX-CUT 문제를 지역최적화를 이용한 Hybrid GA를 사용하여 풀고, 얻은 해의 품질을 multi-start local search 알고리즘과 비교한다. MAX-CUT 문제는 주어진 그래프 $G = (V, E)$ 에 대해 정점들을 두 집합 S_1 과 S_2 로 나누고, S_1 과 S_2 를 잇는 간선의 가중치들의 합을 최대로 하는 문제이다. 주어진 그래프의 크기는 $|V| \leq 1000$, $|E| \leq 10000$ 이며, 구현한 GA 프로그램은 180초 이내에 정답을 출력하여야 한다.

II. GA DESCRIPTION

다음은 이번 프로젝트에서 구현한 GA Algorithm이다. Algorithm은 크게 GA와 MAIN 두 가지 부분으로 구성된다.

Algorithm 1 GA Part

```
1: procedure GA
2:    $P \leftarrow P_{init} \cup P_{random}$ 
3:   repeat
4:      $P_{new} \leftarrow \emptyset$ 
5:     for 1 to  $K$  do
6:        $p_x \leftarrow select(P)$ 
7:        $p_y \leftarrow select(P)$ 
8:        $p_z \leftarrow crossover(p_x, p_y)$ 
9:        $p_z \leftarrow mutation(p_z)$ 
10:       $p_z \leftarrow local\_search(p_z)$ 
11:       $P_{new} \leftarrow P_{new} \cup \{p_z\}$ 
12:    end for
13:     $P \leftarrow replace(P, P_{new})$ 
14:   until 수행시간  $\leq GA_{duration}$ 
15:   return  $P$ 에서 가장 품질이 좋은 해  $x$ 
16: end procedure
```

Algorithm 1은 GA 부분을 서술한 Algorithm이다. 먼저 처음에 N_p 개의 해를 만든 후 해집단 P 에 넣는다. 이 때 해집단 P 는 입력으로 받은 해집단 P_{init} 과, 랜덤으로 만든 해집단 P_{random} 의 합집합이다. 매 세대마다 두 해를 select한 후, crossover와 mutation, local search를 거쳐 새로운 해집단 P_{new} 를 생성한다. 그리고 P 와 P_{new} 로부터 replace를 거쳐 다음 세대의 해집단 P' 을 만든다. GA의 수행시간이 $GA_{duration}$ 이 지나면 반복문을 빠져나오며, 가장 좋은 해를 반환한다.

Algorithm 2 MAIN Part

```
1: procedure MAIN
2:    $P_{init} \leftarrow \emptyset$ 
3:    $GA_{rule} \leftarrow make\_GA\_rule()$ 
4:   for  $(R_i, GA_i)$  in  $GA_{rule}$  do
5:      $P_{tmp} \leftarrow \emptyset$ 
6:     for 1 to  $R_i$  do
7:        $x \leftarrow GA_i(P_{init})$ 
8:        $P_{tmp} \leftarrow P_{tmp} \cup x$ 
9:     end for
10:     $P_{init} \leftarrow P_{init} \cup P_{tmp}$ 
11:  end for
12:   $P_{init}$  에서 제일 좋은 해를 출력
13: end procedure
```

Algorithm 2은 MAIN 부분을 서술한 Algorithm이다. 여기서는 parameter가 다른 GA들을 각각 몇 번 실행할 것인지를 서술하는 rule을 작성한다. 그리고 작성한 rule에 따라 GA를 수행하며, 이전 rule까지의 GA를 돌려 얻은 해들을 이번 rule에 대응하는 GA의 입력으로 전달한다. 최종적으로는 수행한 모든 GA에서 제일 좋은 해를 찾아 출력한다.

A. Chromosome Design

해의 표현은 단순하게 이진 스트링 모양의 염색체로 표현하였다. 그래프 $G = (V, E)$ 를 S 와 S' 으로 나누었다고 가정할 때, 어떤 정점 i 가 S 에 속한다면 0, 아니면 1로 표현하였다. 실제 구현에서는 integer의 배열 표현으로 해를 표현하였으며, 편의상 입력받은 정점들을 0-based index로 바꾼 후 저장하였다.

B. Selection

Selection은 지난 프로젝트와 유사하나, 이번에는 룰렛휠 알고리즘만을 적용한다. 선택압을 조절하는 상수인 k 값은 4로 고정하였다.

C. Crossover

Crossover 역시 지난 과제와 동일하게 uniform crossover를 사용한다. 임계 확률을 p_0 로 설정하였을 때, 두 부모해 s_1, s_2 가 주어지면, 각각의 유전자 위치에 대해 p_0 의 확률로 s_1 의 유전자를, $1 - p_0$ 의 확률로 s_2 의 유전자를 선택한다. p_0 는 0.5로 고정하였다.

D. Mutation

Mutation 또한 지난 과제와 유사하나, 이번에는 임계 확률 p_1 을 0.01로 고정시켰다. 각각의 유전자에 대해 p_1 의 확률만큼 값을 반전시킨다.

E. Replace

Replace는 지난 과제와 완벽하게 일치한다. 기존의 해집단 P 와 새로 만든 해집단 P_{new} 를 합친 후, 품질이 나쁜 해를 cutting하여 다음 세대의 해집단 P' 을 만든다. 이때 P_{new} 에서 품질이 제일 좋은 해는 반드시 P' 에 들어가고, P 에서 품질이 가장 나쁜 해는 반드시 P' 에서 빠지도록 구현하였다.

III. EXPERIMENT

TABLE I: Experiment Environment

CPU	Intel Xeon E5-2650 (8-core 2.00 GHz)
Memory	128GB DDR3 (1,600 MHz)
OS	CentOS Linux 7.2
Compiler	g++ (GCC) 4.8.5

Table I은 구현한 Hybrid GA를 돌린 실험 환경이며, 프로그램은 C++로 작성하였고 single thread만을 사용하도록 구현하였다. 컴파일러는 g++을 사용하였으며, c++11과 O3 최적화 옵션을 적용하였다.

TABLE II: Parameters for GA

rule 1	
repeat	64
duration	1.0s
$N_{population}$	256
$K_{population}$	32
rule 2	
repeat	16
duration	3.0s
$N_{population}$	512
$K_{population}$	32
rule 3	
repeat	3
duration	20.0s
$N_{population}$	1024
$K_{population}$	64

Table II는 실험에서 사용한 상수들이다. 여러 번의 실험을 거쳐 비교적 안정적이고 높은 품질의 해를 얻을 수 있는 상수들을 찾아 세팅하였다.

A. Result

TABLE III: Results on Sample Instances (Hybrid GA)

testset	avg	std	max	min
chimera_946	31856	0	31856	31856
cubic_1000	887	3.9	894	878
planar_800	3060.6	1.6	3063	3057
random_1000	9519.0	5.9	9524	9509
random_500	4743	0	4743	4743
toroidal_800	562	1.4	564	558

Table III은 6개의 샘플 데이터들에 대해 Hybrid GA를 30번 수행하여 얻은 평균, 표준편차, 최댓값, 최솟값을 나타낸

것이다. *chimera_946*과 *random_500* 샘플의 경우 항상 최적값을 찾았다. 지난 프로젝트에서 작성한 순수 GA의 경우 이 샘플들에 대해 최적해를 찾지 못하는 경우가 많았다는 점에서 해의 품질이 상당히 올라갔음을 확인할 수 있다. 반면 *cubic_1000* 샘플의 경우 최적해를 찾지 못하는 경우가 많았다.

TABLE IV: Results on Sample Instances (Multi-start local search)

testset	num	avg	std	max	min
chimera_946	1849405	29876.7	224.2	30256	29440
cubic_1000	1887560	741.1	4.64	754	732
planar_800	1438167	2972.0	2.26	2977	2967
random_1000	749513	9292.97	11.77	9320	9276
random_500	1784857	4690.4	6.45	4703	4678
toroidal_800	2686490	473.27	2.94	484	470

Table IV은 6개의 샘플 데이터들에 대해, num개의 해를 생성하는 Multi-start local search를 30번 수행하여 얻은 평균, 표준편차, 최댓값, 최솟값을 나타낸 것이다. 모든 경우에 대해서 Hybrid GA에 비해 낮은 품질의 해를 얻는 것을 확인할 수 있었다.

B. Analysis on one GA run

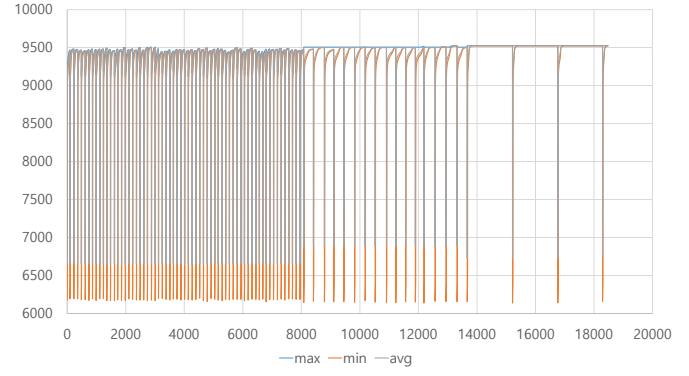


Fig. 1: 20000세대 동안의 품질 변화

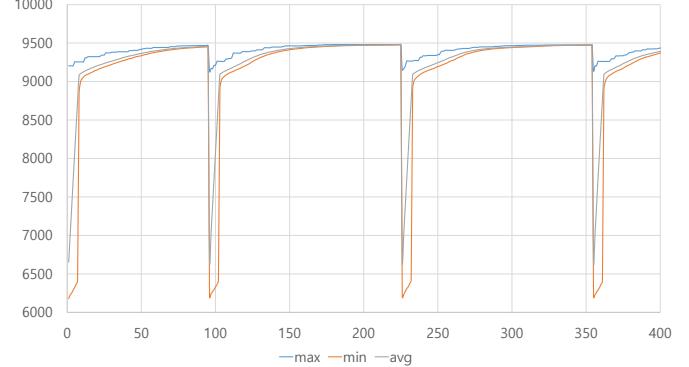


Fig. 2: Rule 1에서의 품질 변화

Figure 1은 *random_1000* 샘플에서 GA를 한 번 수행하였을 때 세대별 품질 변화를 나타낸 것이다. 그래프에서 확인할 수 있듯이 수렴은 매우 빠르게 일어나며, 이는 Figure

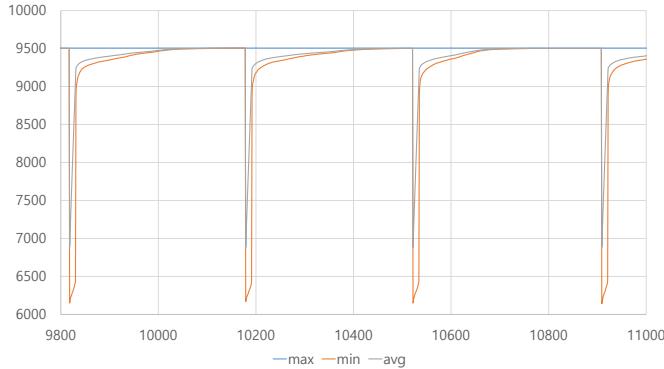


Fig. 3: Rule 2에서의 품질 변화

2와 Figure 3에서도 확인 가능하다. Rule 1의 경우 $P_{init} = \emptyset$ 이므로 max값이 바뀌는 한편, Rule 2부터는 max 값은 거의 변하지 않고, min값과 avg값이 빠르게 max값으로 다가가는 것을 관찰 가능하다.

IV. DISCUSSION

Hybrid GA를 구현하면서 최종 해의 높은 품질을 얻는 데 어려움을 겪었던 부분들은 다음과 같다.

A. 초기 해 생성

Hybrid GA 역시 초기 해가 최종 해의 품질에 큰 영향을 끼쳤다. Local Search로 인해 수렴이 더 빠르게 일어나므로, 그만큼 초기해를 다시 생성하여 GA를 수행하는 것이 높은 품질의 해를 얻는데 도움이 많이 되었다. 다만 쉬운 문제의 경우 초기 해를 여러 번 다시 생성하다보면 운좋게 최적해가 나왔으나, 어려운 문제의 경우 높은 품질의 해에서 최적해로 다가가는데 GA의 도움이 필요하였다. 따라서 이번 프로젝트에서는 180초의 시간제한 중 마지막 60초는 GA를 오래 돌리면서 최적해를 찾고자 노력하였다.

B. 해집단의 다양성 확보 어려움

프로젝트 1과 마찬가지로 해집단의 다양성을 확보하는 것 이 어려웠다. Crossover와 Mutation을 많이 수정해보았지만 최종 해의 품질이 좋아지는 것은 상당히 보기 힘들었다. 최종 해의 품질이 올라간 것은 Local search로 인한 요소가 지배적이었으며, GA를 수정하여 얻는 이득은 관찰하기 어려웠다.

V. CONCLUSION

이번 Project에서는 Local Search를 이용한 Hybrid GA를 이용하여 MAX-CUT 문제를 해결하였으며, 순수 GA에 비해 상당히 높은 품질의 해를 얻을 수 있었다. 하지만 초기해가 끼치는 영향이 매우 크고, 해집단의 다양성 확보가 어려웠다는 점은 프로젝트 1에 비해 아직 개선되지 않았다. Local Search가 상당히 강력한 방법이므로 이를 GA와 잘 연동시켜 서로 시너지 효과를 내는 것을 기대하였지만, GA 부분을 충분히 개선하지 못하여 더 높은 품질의 해를 얻지 못하였던 것 같다. 프로젝트 1에 비해 해의 품질이 많이 올라가 뿐만 아니라, 본인의 GA architecture 설계 능력에 대해 좌절감을 느낄 수 있는 프로젝트였다.