2018 GA Project 1: MAX-CUT

due: 4월 20일(금) 오후 11시 59분

1. Problem Statement

방향성이 없는 간선에 가중치가 부여된 단순 그래프(simple graph) $G=(V,E)$가 주어진다. V와 E는 각각 정점 집합과 간선 집합이다. 정점 집합 V를 두 개의 집합 S와 $S'(=V-S)$로 나누는 방법은 여러 종류가 있는데, 여러분은 S와 S'를 잇는 간선의 가중치의 총합을 최대화하도록 나누어야 한다.

그림 1은 10개의 정점과 14개의 간선으로 이루어진 단순 그래프의 한 인스턴스다. 각 간선의 가중치는 1 또는 -1로 주어져있다. 그림 2는 그림 1의 그래프의 정점 집합을 둘로 나누는 방법 중 하나로, $S=\{1,6,8,9\}$, $S'=\{2,3,4,5,7,10\}$로 나눈 것이다. S와 S'에 속한 정점들을 각각 검은색과 흰색으로 표시하였다. 굵게 표시된 간선은 S와 S'를 잇는 간선으로, 가중치의 총합은 8이 된다. 그림 3은 정점 집합을 둘로 나누는 다른 방법으로, 두 집합을 잇는 간선의 가중치의 총합은 7이다.

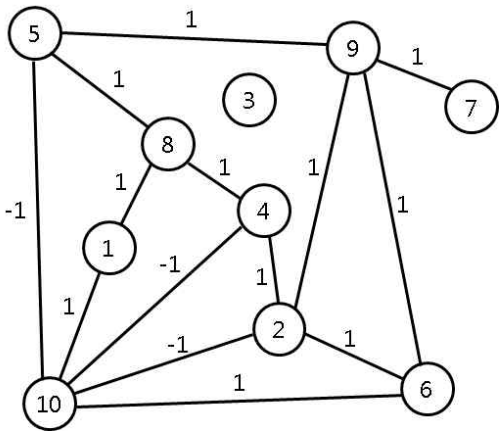

그림 1

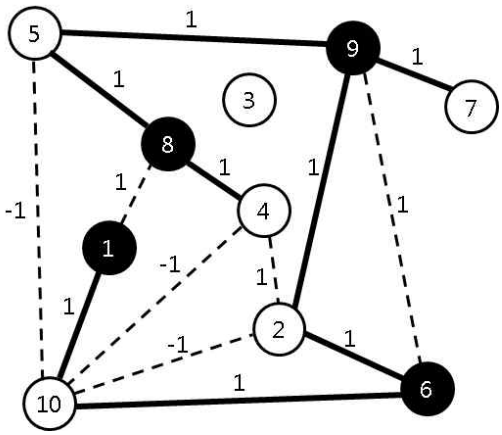

그림 2

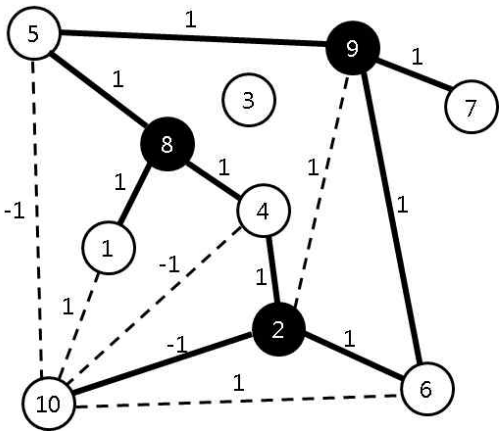

그림 3

주어진 단순 그래프 G에서 정점 집합을 두 집합 S와 $S'(=V-S)$로 나눌 때, 이 둘을 잇는 간선의 가중치의 총합을 제한 시간 내에 가능한 최대화하는 Genetic Algorithm을 작성하라.

- 인스턴스의 크기: 정점의 개수 $|V| \leq 500$, 간선의 개수 $|E| \leq 5000$
- 제한 시간: 한 인스턴스 당 180초

[입력]
입력 파일의 이름은 **maxcut.in**이다. 입력 파일의 각 항목은 공백과 줄바꿈으로 구분된다. 첫 줄에는 정점의 개수 $|V|$와 간선의 개수 $|E|$가 주어진다. 둘째 줄부터 차례대로 각 간선의 양 끝 정점의 번호와 간선의 가중치가 주어진다. 정점의 번호는 1 이상 $|V|$ 이하의 정수로 주어지고, 간선의 가중치는 정수로 주어진다. 시작점과 끝점이 같은 간선(self-loop)이나 동일한 두 정점을 잇는 복수 개의 간선(multiple edges)은 주어지지 않는다.

[출력]
출력 파일의 이름은 **maxcut.out**이다. 여러분이 구한 최적 집합 S 내의 모든 정점 번호를 한 줄에 공백으로 구분하여 출력한다. 출력 파일에 가중치 총합 수치를 포함하지 않는다.

[입출력 예]
입력 (그림 1의 그래프)

```
10 14
1 10 1
1 8 1
2 10 -1
2 9 1
2 6 1
2 4 1
4 8 1
4 10 -1
5 9 1
5 10 -1
5 8 1
6 10 1
6 9 1
7 9 1
```

출력 (그림 2의 $S = \{1, 6, 8, 9\}$)

```
1 6 8 9
```

2. Restriction

이번 과제에서는 순수 GA만 사용하도록 한다. 지역 최적화(local optimization)의 성격을 갖는 과정이 어떤 식으로든 포함되지 않도록 한다. 이번 과제는 순수 GA가 갖는 한계를 경험하는 것을 목표로 하며 이를 위해 crossover, mutation 및 기타 preprocessing에서 지역 최적화 과정을 금지하는 것으로 이해하면 된다.

이번 과제를 포함, 이 강좌의 모든 과제에서 제출할 프로그램은 반드시 CPU 상에서 single thread로 수행되어야 한다. 멀티코어 혹은 멀티쓰레드 컴퓨팅은 허용하지 않으며, GPU 활용 역시 허용하지 않는다. 프로그램은 외부와 통신 없이 단독으로 수행되어야 한다. 앞의 제약을 위배한 경우 큰 페널티를 받는다.

3. Language

프로그램 언어는 C, C++, JAVA 중 하나를 사용한다. 다른 언어를 사용하는 경우 미리 조교와 상의하여 승인된 경우에 한해 허용한다. 제한 시간(180초)을 두고 수행하는 채점 방식 상, 수행속도가 빠른 C, C++를 사용하는 것을 권장한다. C, C++나 JAVA가 기본적으로 제공하는 standard library 이외의 다른 라이브러리를 사용하는 경우에는 미리 조교와 상의해야한다. 조교와 상의 없이 다른 언어나 라이브러리를 사용하는 경우, 큰 페널티를 받을 수 있다.

4. Compilation and Execution

제출 시 반드시 Makefile을 포함해야 하며 **make all** 명령으로 컴파일이 가능하도록 만들어야 한다. C, C++를 사용하는 경우 컴파일러의 최적화 옵션은 -O3까지 사용할 수 있다. 또한 **make run** 명령으로 프로그램이 수행되도록 한다. 아래 예시와 같이 명령을 입력했을 때, 여러분이 구한 해가 maxcut.out에 기록되어야 한다.

```
$ ls
Makefile yourfile1.cpp yourfile2.cpp ...
$ make all                                  ▷ 컴파일
...
$ cp /home/ga_ta/instance1 ./maxcut.in      ▷ 사용할 인스턴스 복사
$ make run                                  ▷ 프로그램 수행
...
$ cat ./maxcut.out                          ▷ 수행 후 생성된 파일 출력
1 6 8 9
```

컴파일과 수행은 리눅스에서 가능해야 한다. 채점 컴퓨터는 Intel(R) Core(TM) i7 870 2.93GHz이며 OS는 Ubuntu 12.04 LTS이다. (컴퓨터의 사양 차이를 고려하여 알고리즘을 작성하는 것을 권장한다.)

5. Report

프로젝트에 대해 보고서를 작성하여 제출한다. 분량은 A4 8페이지 이하이며, 논문 형식으로 작성한다. 보고서 내에 소스코드를 그대로 포함해서는 안 된다. 보고서 본문에는 최소한 다음 내용을 포함시킨다.

1. 사용한 GA의 구조
2. 해의 표현 (chromosome design)
3. 사용한 연산자에 대한 설명 (selection, crossover, mutation, replacement 등)
4. 강좌 페이지에 공개된 다섯 개의 샘플 인스턴스에 대해서 GA를 각각 최소 30번씩 수행하여 가장 좋은 결과, 평균 결과, 표준편차를 테이블로 기록 (최소 150번의 GA run)
5. 위에서 수행한 GA run들 중 하나를 선택, 세대 진행에 따른 population 분석 (해들의 평균 품질, 최고 품질 등)
6. Discussion (느낀 점, 잘 안 되는 점, 의외의 현상, 예상대로 된 점 등)

6. Submission

학번을 파일명으로 하여 zip 또는 tar(+gz) 형식으로 압축한다. 압축 파일에는 소스코드와 Makefile, 보고서(pdf)가 포함되어야 한다. 그 후, 자신의 학번을 제목으로 하여 조교에게 이메일(ga_ta@soar.snu.ac.kr)로 제출한다. 메일 내용에는 자신의 학번과 이름을 반드시 포함한다. 첨부파일은 압축파일 하나만 있어야 하며, 첨부파일이 없거나 둘 이상 있으면 제출로 인정하지 않는다.

7. Grading

보고서와 프로그램이 각각 1:1의 비율로 반영된다. 둘 모두 상대평가에 의해 점수가 부여되며, 특히 프로그램의 경우에는 여러분의 프로그램이 구한 해의 품질에 따라 점수 차이가 크게 벌어질 수 있다. 단, 첫 번째 과제의 경우에는 순수 GA의 특성을 감안하여 기본 점수의 비중을 높게 부여한다. 이후 과제에서는 점수 차이가 늘어날 예정이다.

프로그램은 별도로 준비된 테스트 인스턴스에 대해서 출력하는 해의 품질에 따라 평가된다. 각 인스턴스마다 프로그램은 한 번 수행된다. **프로그램이 제한 시간을 초과하여 수행된 경우, 또는 출력의 형식이나 내용이 잘못되어 채점이 불가능한 경우에는 해당 인스턴스에 대해서 0점을 받게 된다.** 각 인스턴스에 대해서 가장 좋은 품질의 해가 만점을 받게 되며, 다른 해는 그 품질의 차이에 따라 부분점수를 받게 된다. 부분점수는 품질 차이의 절대치와 상대치(비율)에 따라 부여된다. 외부 프로그램도 경쟁에 포함될 수 있다.

보고서는 위의 최소 내용을 모두 포함할 경우 기본 점수가 부여되며 나머지 성적은 보고서 내용이 충실한 정도에 따라 평가한다. 보고서에 불필요한 내용(표지, 소스코드, 별도의 분석 없는 raw data 등)을 포함하는 경우 감점요인이 된다.

- 타인의 source code는 절대 보지 말 것. 자동 Copy detection program에 의하여 카피를 판별한 후 원본과 복사본 공히 0점 처리한 후 최종 학점에서 한 grade 강등함. 일부든 전부든 모두 카피로 처리함. Source code copy 후 변경도 detection 됨. 선배들의 source code도 비교 대상임.
- 수강생이 많은 관계로 프로그램 채점의 많은 부분이 자동적으로 이루어질 예정이다. 수강생의 실수로 인해 예외적인 상황이 발생하여 전체 채점이 지연될 경우 큰 감점 요인이 될 수 있다.
- 과제물을 늦게 제출한 경우, 조교에게 최종적으로 도착한 시간을 기준으로 매 24시간마다 20%씩 감점된다. 채점 및 규정 적용은 제일 마지막 제출물에 따라 처리된다. 보고서와 프로그램을 따로 제출할 수는 없다.