2015 GA Project 1: Traveling Salesman Problem

2015. 4. 2
due: 4월 20일 오후 11시 59분

과제 개요

물류 회사의 운전기사인 김씨는 배달을 위해 n개의 지점을 순회해야 한다. 각 지점의 위치는 이차원 실수 좌표 $(x,\ y)$로 주어지고$(0 \le x,\ y \le 100)$, 두 위치 $(x_1,\ y_1)$, $(x_2,\ y_2)$ 사이의 거리는 실수 $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$으로 계산된다. 각 지점의 좌표는 모두 다르다. n개의 지점 중 한 곳을 선택하여 출발하여 나머지 지점을 모두 방문한 후 시작 지점으로 되돌아오는 순환 경로 (Hamiltonian Cycle) 중 가능한 짧은 것을 찾으려 한다. Genetic algorithm을 이용하여 가능하면 짧은 경로를 찾으라.

프로그램은 C, C++, JAVA 언어를 사용한다. 컴파일러의 최적화 옵션은 -O3까지만 사용할 수 있다.

시간제한과 Test 기기 환경

테스트는 제한된 시간 동안 이루어지며, 제한은 입력으로 주어질 것이다. 여러분의 프로그램은 이 시간동안 최대한 좋은 해를 찾아서 출력해야 한다. 만일 프로그램이 제한 시간을 넘겨 수행되면(real time 기준) 해당 입력에 대한 점수는 없다.

프로그램이 수행될 서버는 Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz를 사용할 예정이다. 프로그램이 계산에 사용하는 자원은 하나의 CPU core로 제한한다. 즉, 멀티쓰레딩 혹은 GPU를 활용하는 병렬화 등을 모두 금지한다. 또한 프로그램이 사용할 수 있는 메모리 크기는 512MB 이하이다. 이는 별 문제가 없는 부분이지만 혹시 사용할지 모르는 특이한 방법을 금지하기 위함이다. 제출한 프로그램이 위와 관련하여 문제가 생길 소지가 있다면 이를 보고서에 명시하도록 한다.

GA의 구조 제한

고객의 수 n은 1차 프로젝트에서는 318 이하로 주어질 것이다. 1차 프로젝트에서는 순수 GA의 틀을 벗어나면 안된다. 지역 최적화(Local Optimization) 과정이 명시적으로 포함되어서는 안되고, 교차나 변이 등에서도 지역 최적화 성격의 작업이 포함되면 안된다. 이를 명확하게 정의하는 것은 어려운데, 기본적으로 교차나 변이 과정에서 해의 품질을 계산하거나 활용할 수 없다고 생각하면 된다. 물론 선택이나 대치에서는 해의 품질을 계산하는 것이 얼마든지 가능하다. 이것은 선택이나 대치에서 지역 최적화 연산을 할 수 있다는 의미가 아니다. 선택 연산에서는 해의 품질을 서로 비교하여 그 중 좋은 해를 return하는 과정이 필요할 수 있기 때문에 해의 품질 계산이 허용되는 것이고, 대치 연산에서는 대치할 해를 찾기 위해서 가장 품

질이 나쁜 해를 찾는 과정이 필요할 수 있기 때문에 해의 품질 계산이 허용되는 것이다. 초기 해를 만드는 과정에서도 지역 최적화 성격의 작업이 들어가서는 안된다. 지점간의 거리 등을 활용한 유전자 재배치도 물론 할 수 없다. 만일 지역 최적화 성격의 작업이 들어간 GA를 제출한다면, 이것은 과제의 가장 기본적인 조건을 어긴 것이므로 프로그램 점수를 최대 50%만 부여한다.

1차 프로젝트는 순수한 형태의 GA가 갖는 한계를 경험해 보라는 것이므로 결과에 대해 실망감을 좀 느껴도 좋다. 그렇지만 순수 GA로 낼 수 있는 최대한의 품질은 내도록 노력하라. 과제물의 프로그래밍 점수는 여러분이 제출한 GA가 구한 해의 품질에 따라 주어진다.

입출력 양식

[입력]
입력 파일의 이름은 cycle.in 이다. 첫 번째 줄에는 지점의 수 $n(1 \leq n \leq 318)$이 주어진다. 그 다음 n개의 줄에는 각 지점의 (x, y)좌표가 주어진다. 좌표는 공백으로 분리되어 주어지며, 소수점 아래 둘째 자리까지 주어질 수 있다. 마지막 줄에는 이 입력에 대해서 여러분이 사용할 수 있는 시간이 몇 초인지를 나타내는 실수가 주어진다.

[출력]
출력 파일의 이름은 cycle.out 이다. 첫째 줄에 자신이 제시하는 경로를 출력한다. 경로는 방문한 순서대로 지점의 번호를 출력하도록 하며, 지점의 번호는 입력에서 주어진 순서대로 1부터 n까지이다.

[입력 예]

```
6
4 5.0
0.00 0
4.0 3.0
0 4
2.51 5.0
1.0 5.00
10.00
```

[출력 예]

```
1 5 6 4 2 3
```

(※이 경로의 길이는 약 15.4142다.)

보고서 및 Source Code 제출 방법

먼저 게시판을 통해 몇 개의 예제 입력이 제공될 것이다. ($n = 11, 21, 51, 101$) 보고서에는

기본적으로 제공된 데이터에 대한 실험 결과를 모두 수록하도록 하며, 필요에 따라 추가적인 실험을 개인적으로 수행하여도 좋다. 이와 별도로 다양한 입력에 대해 여러분이 제출한 source code를 컴파일하여 명시된 수행 시간 범위에서 확인을 할 것이다. 다양한 문제에 대한 적응력을 보려면 비슷한 문제를 직접 여러 개 만들어 확인해보는 것이 좋다.

과제는 보고서(pdf, docx, hwp만 허용)와 소스 코드 모두 ga_ta@soar.snu.ac.kr로 제출한다. 소스 코드와 보고서의 due date는 같으므로 유의하여야 한다.

보고서는 아래의 내용을 반드시 포함해야 하며, A4 여덟 페이지 이하의 논문 형식으로 작성한다.

1. 사용한 GA의 구조
2. 사용한 연산자에 대한 설명(Selection, Crossover, Mutation, Replacement)
3. 문제의 표현(chromosome design)
4. 제공된 입력에 대해서 GA를 각각 최소 30번씩 수행하여, "가장 좋은 결과," "평균 결과," "표준편차"를 한 테이블에 기록하고, 실험 환경 및 사양을 명시할 것.
5. 실험한 데이터 중 $n = 101$인 데이터에 대해 대표적인 run을 하나 정하고, 세대가 진행됨에 따라 population 내의 해들의 평균 품질과 최고 품질의 모양을 plotting 할 것.
6. Discussion(느낀 점, 잘 안되는 점, 의외의 현상, 예상대로 된 점, 등등)

소스 코드 제출을 위해서 먼저 자신의 학번으로 디렉토리를 만들고, 그 안에 컴파일에 필요한 모든 파일과 Makefile을 담도록 한다. 설치가 필요한 외부 라이브러리(boost 등)는 원칙적으로 사용할 수 없으며, 반드시 필요한 경우에는 컴파일에 필요한 모든 파일을 첨부하고 관련 사항을 보고서에 기재하도록 한다. 첨부한 Makefile을 이용하여 컴파일과 실행을 모두 해볼 수 있어야 하며, 리눅스 환경에서 문제없이 구동되어야 한다. Makefile은 최소한 다음의 세 가지 내용을 포함해야 한다.

1. make all : 실행에 필요한 파일들을 모두 생성한다. 컴파일과 관련된 모든 과정이 이를 통해 이루어져야 한다. 여러 파일이 필요한 경우 의존성 관계를 마음대로 넣을 수 있으며, 컴파일 최적화 옵션은 -O3 까지만 사용 가능하다.
2. make run : 프로그램을 실행한다. 생성된 실행 파일을 실행하는 내용을 포함해야 하며, 현재 디렉토리(.)가 path에 잡혀있지 않을 수 있음에 유의한다. 프로그램이 실행되면 cycle.in에서 입력을 읽고, 제한된 시간동안 프로그램이 동작된 후, cycle.out에 결과를 기록한 후 자동으로 종료되어야 한다.
3. make clean : 생성된 파일을 모두 지운다. 채점시 프로그램을 실행할 경우 환경 변화가 있을 수 있으므로, 컴파일 작업을 다시 수행하게 된다. 이 과정이 원만하게 이루어질 수 있도록 필요한 파일을 삭제한다.

채점시에는 차례로 make clean, make all, make run을 수행하게 된다. 세 과정 중 하나라도 정상적으로 수행되지 않을 경우 감점 요인이 된다. 시간 측정은 time make run의 형태로 이루어질 예정이므로 참고하도록 한다.

위와 같이 코드와 Makefile을 정리한 후, 학번을 파일명으로 하여 zip 또는 tar(+gz) 형식으로 압축한다. 그 후, 자신의 학번을 제목으로 하여 조교에게 이메일로 제출한다. 메일 내용에는 자신의 학번과 이름을 반드시 포함한다.

채점 방식

기본적으로 보고서와 프로그램이 각각 1:1의 비율로 반영된다. 둘 모두 상대평가에 의해 점수가 부여될 수 있으며, 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.

보고서는 앞에서 언급한 필수적인 내용을 모두 갖춘 경우 기본 점수를 받게 된다. 좋은 실험 내용이나 논의사항이 포함된 경우, 각각의 품질에 따라 추가 점수를 받을 수 있다. 역으로, 내용이 부실한 경우 등에는 그 정도에 따라 감점이 있을 수 있다. 보고서는 우리말로 작성하는 것을 원칙으로 한다. 위와 같이 채점한 후, 가장 우수한 보고서가 만점이 되도록 점수가 조정된다.

프로그램은 준비된 여러 개의 테스트 데이터에 대해서 여러분의 프로그램이 구한 해의 품질에 따라 평가된다. 각 케이스마다 프로그램은 한 번 수행된다. 프로그램이 제한 시간을 초과하여 수행된 경우, 출력의 형식이나 내용이 잘못된 경우에는 그 케이스에 대해서 0점을 받게 된다. 그 외의 경우에는 각 케이스에 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.

주의사항

* 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.
* 절대로 보고서에 30회의 수행 결과를 일일이 기록하지 말 것. 평균, 표준편차 등의 통계치로 충분함. 쓸데없는 detail로 보고서를 채울 경우 감점 요인이 됨.
* 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. 이메일이 반송된 경우, 첨부 파일이 누락된 경우, 각종 파일의 형식이나 이름이 맞지 않는 경우 등에는 다시 제출하는 것이 가능하다. 이때에는 위의 딜레이 규정에 따라 늦게 제출한 과제물로 처리될 수 있다. 여러 번 제출하는 것이 가능하며, 채점 및 규정 적용은 제일 마지막 제출물에 따라 처리된다. 보고서와 프로그램을 따로 제출할 수는 없다.