Homework #7

Search Tool (Version 1)

AI Programming

과제 기간: 2022.10.31(월) ~ 2022.11.13(일) 밤 11:50까지 // 2주간 진행

\*참고: Homework #6번은 없습니다!

* 이번 과제부터는 Colab을 사용할 필요 없습니다.
* 작성한 소스코드 및 문제(problem) 텍스트파일이 포함된 폴더 전체를 압축해서 제출하세요. 참고: 문제에 해당하는 txt 파일은 “problem/” 폴더 아래에 저장하세요.
* 구현이 완료되면 주어진 6개 문제를 대상으로 프로그램을 실행하여 구현이 정확한지를 확인하세요. 6개의 문제를 실행하고, 터미널 출력 화면을 캡쳐하여 본 문서에 붙여넣기 한 후, WORD 파일도 같이 제출하세요 (마지막 페이지 참고).

**Overview**

We want to solve numerical optimization problems (n) and travelling salesperson problems (tsp) using search algorithms. In this assignment, you are provided with some program codes that implement steepest-ascent hill climbing and first-choice hill climbing algorithms for solving both types of problems, but with many of the functions left incomplete. Your job is to complete the programs by filling up the codes for those functions (indicated by ‘###’ after the function name), and then put some of the functions together in library modules (called ‘numeric’) to minimize code duplications. A typical outcome of numerical optimization is shown in the shaded box below.

**<Numerical optimization problem>**

Enter the file name of a function: problem/Convex.txt

Objective function:

(x1 - 2) \*\* 2 +5 \* (x2 - 5) \*\* 2 + 8 \* (x3 + 8) \*\* 2 + 3 \* (x4 + 1) \*\* 2 + 6 \* (x5 - 7) \*\* 2

Search space:

x1: (-30.0, 30.0)

x2: (-30.0, 30.0)

x3: (-30.0, 30.0)

x4: (-30.0, 30.0)

x5: (-30.0, 30.0)

Search algorithm: First-Choice Hill Climbing

Mutation step size: 0.01

Solution found:

(2.005, 5.001, -8.003, -0.997, 7.003)

Minimum value: 0.000

Total number of evaluations: 24,358

**Sample Problems to Be Solved**

You are also provided with six text files for problems to solve, where each contains the specifics of the sample problem to be solved.

The first three files contain numeric functions to be minimized: Convex function, Griewank function, and Ackley function. Each file in its first line contains the function expression written in Python syntax assuming that your program will import ‘math.py’. You can use the Python ‘eval’ function to evaluate the expression, after assigning values to the variables using the ‘exec’ function. The rest of the lines each contains a variable name, its lower bound, and its upper bound. All the functions are five dimensional. You should make sure that the next point you move to during search is always within the search space (i.e., you must not go out of the lower and upper bounds).

The next three files contain different versions of TSPs with the numbers of cities of 30, 50, and 100. The first line of each of these files contains the number of cities, and the rest of the lines contain city locations represented as coordinates in a 100 × 100 square.

**Implementation Using Modules**

Notice that there are two versions of each algorithm, one for numerical optimization problem and the other for TSP. Since there will be many code duplications among different versions, you should create modules to hold those codes and let them be imported for use. You are recommended to create two modules: ‘numeric’ module for the functions specialized to numeric optimization problems and ‘tsp’ module for those specialized to TSPs.

As shown in the shaded box above, you should query the user to get the name of the file containing the specifics of the problem to be solved. After solving the problem, you should print out messages that describe what kind of problem you solved, which search algorithm you used, how the parameters were set for the search algorithm, and the result of search together with the cost of the optimization, i.e., the total number of evaluations taken by the search algorithm to solve the problem. The following is a typical outcome after solving a TSP

**<TSP problem>**

Enter the file name of a TSP: problem/tsp50.txt

Number of cities: 50

City locations:

(1, 7) (14, 92) (45, 97) (17, 60) (22, 44)

(4, 38) (13, 73) (79, 68) (76, 95) (62, 14)

(25, 75) (26, 9) (88, 81) (56, 65) (64, 71)

(92, 20) (7, 20) (8, 20) (61, 39) (17, 11)

(10, 40) (18, 72) (89, 72) (58, 25) (57, 57)

(66, 70) (36, 72) (89, 91) (18, 90) (72, 49)

(82, 38) (22, 26) (36, 56) (23, 44) (45, 45)

(7, 27) (84, 6) (32, 78) (0, 29) (64, 63)

(45, 24) (21, 81) (37, 16) (86, 57) (65, 99)

(25, 53) (98, 24) (83, 81) (50, 5) (58, 80)

Search algorithm: Steepest-Ascent Hill Climbing

Best order of visits:

34 40 18 9 36 15 22 43 30 29

39 24 25 14 47 27 12 8 44 2

1 28 41 6 3 21 10 37 49 26

13 7 46 23 48 42 17 19 16 35

38 0 11 5 20 31 4 33 32 45

Minimum tour cost: 890

Total number of evaluations: 1,888

Q 1. [problem/Ackley.txt : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |

Q 2. [problem/Convex.txt : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 |

Q 3. [problem/Griewank.txt : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 |

Q 4. [problem/tsp30 : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 |

Q 5. [problem/tsp50 : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |

Q 6. [problem/tsp100 : 실행 결과(터미널 또는 콘솔 출력 화면)를 캡쳐 하여 아래에 첨부하세요]

|  |
| --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 |

\*\* 지금 보고 있는 WORD 문서에 캡처 화면을 첨부하고 소스코드 전체 폴더를 압축하여 PLATO에 제출하세요.

THE END