AI 프로그래밍

#HOMEWORK 10



학번: 201724461

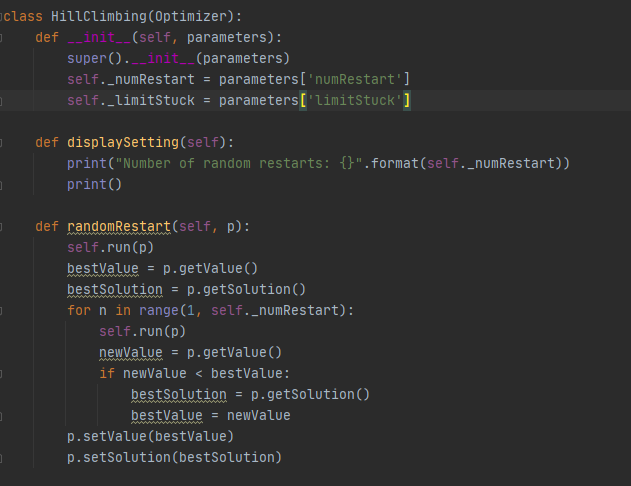
이름: 류지환

Expanding to ‘Optimizer’ Class

1. Optimizer.py
2. Optimizer class
   * 1. Class 개요:

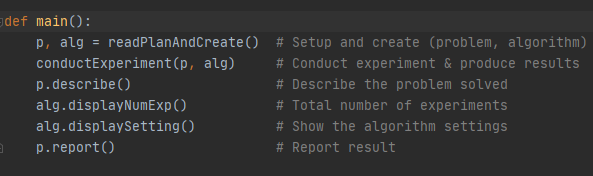


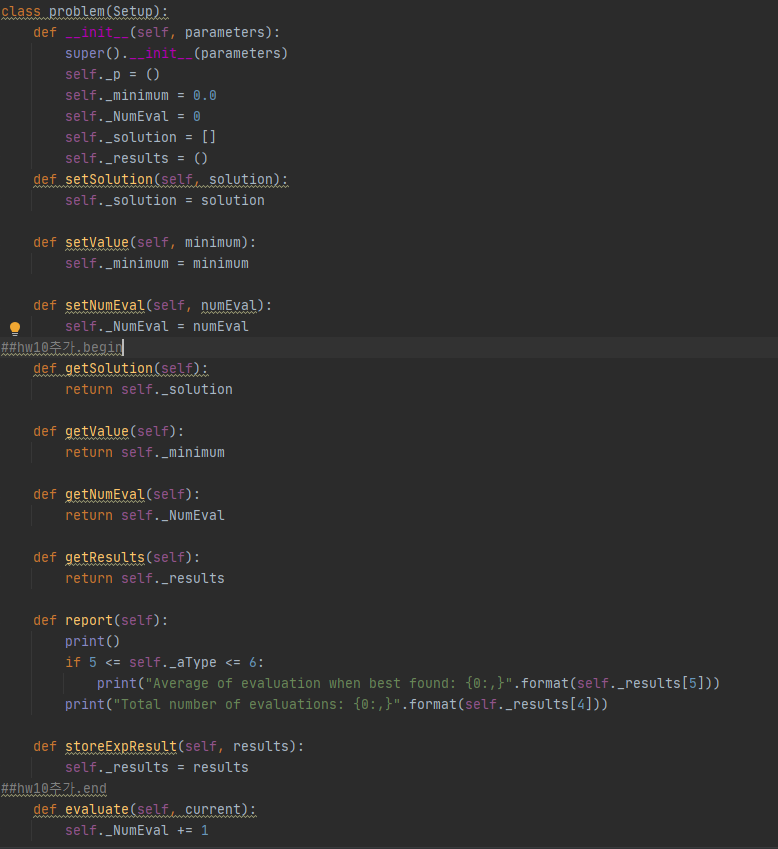
* + 1. 주요 고려 사항:
       1. Def run(self, p): Main 함수에서 numExp 또는 numRestart에 따라 알고리즘을 반복시켜 run 해야 하기에 run 멤버 함수에서 problem 변수 p를 매개변수로 받아 p의 numEval을 0으로 초기화하는 구문을 추가 후 하위 class들에 상속.
  1. HillClimbing(Optimizer) class
     1. Class 개요:



* + 1. 주요 고려 사항:
       1. Def randomRetart(self, p): main 함수에서 numRestart에 따라 알고리즘을 반복시켜 run하고 반복 시마다 bestValue 및 bestSolution을 업데이트, 이를 하위 4개의 class, steepestAscent, firstChoice, stochastic, gradientDescent class에 상속.

New User Interface

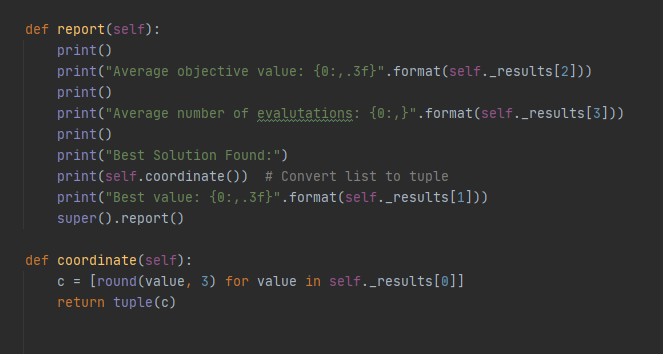
1. Main.py
   1. 함수 개요:
      1. 
2. Problem.py
   1. Problem class
      1. Class 개요:



* + 1. 주요 고려 사항:
       1. Self.\_results:
          1. New user interface에서 출력할 데이터들을 저장하는 멤버 변수 results 선언. (results = (bestSolution, bestMinimum, avgMinimum,  
              avgNumEval, sumOfNumEval, avgWhen))
       2. storeExpResults(self, results):
          1. Results의 값을 외부로부터 받아 저장하는 함수.
       3. Report(self):
          1. Results의 데이터로 결과 출력문 구성.
  1. Def report(self):
     1. TSP:
        1. 함수 개요:

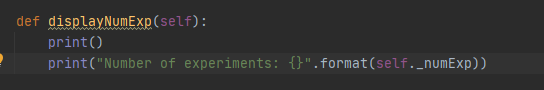


* + - 1. 주요 고려 사항:
         1. Report에 New user interface의 출력문 구성.
         2. 기존에 여러 멤버 변수에서 참조했던 데이터들을 interface에 맞게 results 변수에 저장된 값을 사용.
    1. numeric:
       1. 함수 개요:

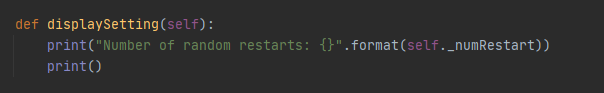


* + - 1. 주요 고려 사항:
         1. Report에 New user interface의 출력문 구성.
         2. 기존에 여러 멤버 변수에서 참조했던 데이터들을 interface에 맞게 results 변수에 저장된 값을 사용.

1. Optimizer.py
   1. Def displaySetting(self):
      1. Optimizer:
         1. 함수 개요:



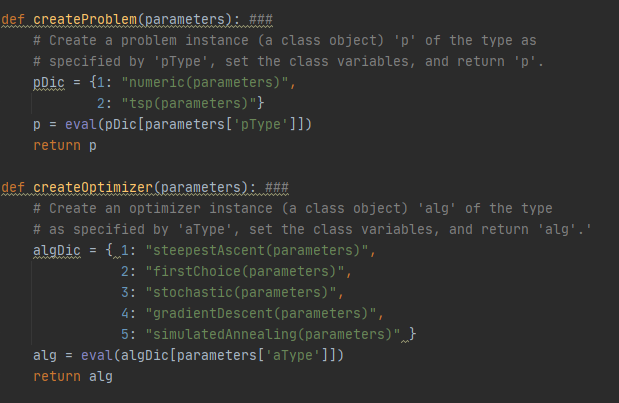
* + - 1. 주요 고려 사항:
         1. 새로운 멤버 변수 numExp를 출력할 함수 구현.
    1. HillClimbing(Optimizer):
       1. 함수 개요:



* + - 1. 주요 고려 사항:
         1. 새로운 멤버 변수 numRestart를 출력할 출력문 구현.
    1. HillClimbing의 sub class(firstChoice, steepestAscent, gradientDescent, stochastic):
       1. 주요 고려 사항:
          1. 각 sub class의 출력문에 맞게 displaySetting 함수를 구현

Changes to the Main Program

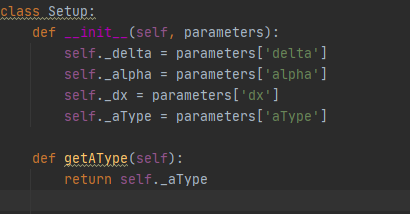
1. CreateProblem(parameters) & createOptimizer(parameters)
   1. 함수 개요:



* 1. 주요 고려 사항:
     1. Exp.txt를 참고해 problem과 optimizer의 모든 case들의 생성자를 각 query와 대응되는 dictionary 형태로 구성.

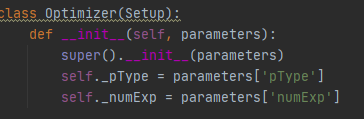
Changes to the Class Hierarchy

1. Setup.py
   1. Setup class
      1. Class 개요:

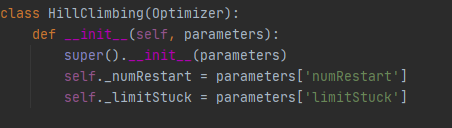


* + 1. 주요 고려 사항:
       1. New user interface에 요구되는 멤버 변수들 중 delta, alpha, dx, aType를 최상위 클래스에 선언, 이를 생성자에서 parameters를 매개 변수로 받아 초기화.

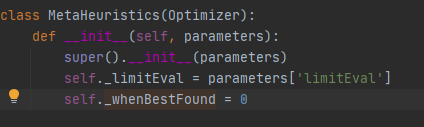
1. Optimizer.py
   1. Optimizer(Setup) class
      1. Class 개요:



* + 1. 주요 고려 사항:
       1. New user interface에 요구되는 멤버 변수들 중 pType, numExp를 Optimizer 클래스에 선언, 이를 생성자에서 parameters를 매개 변수로 받아 초기화.
  1. HillClimbing(Optimizer) class
     1. Class 개요:

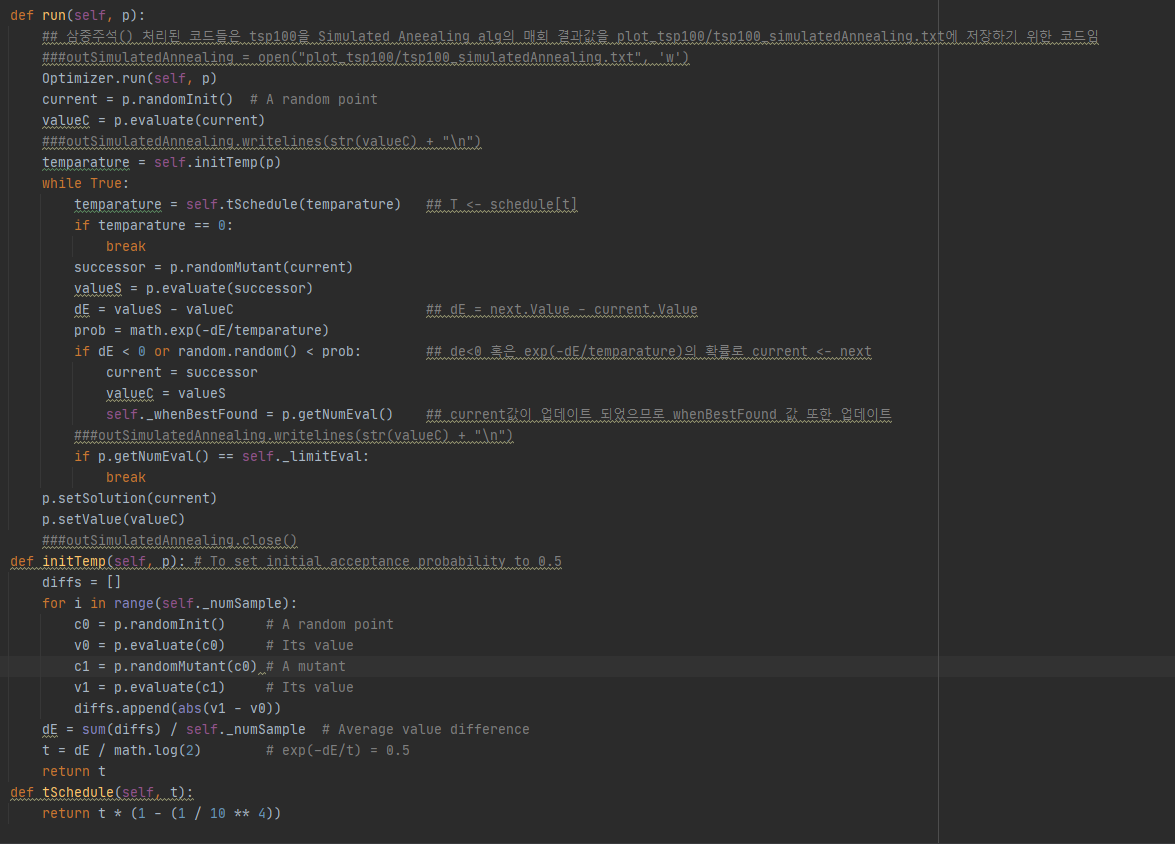
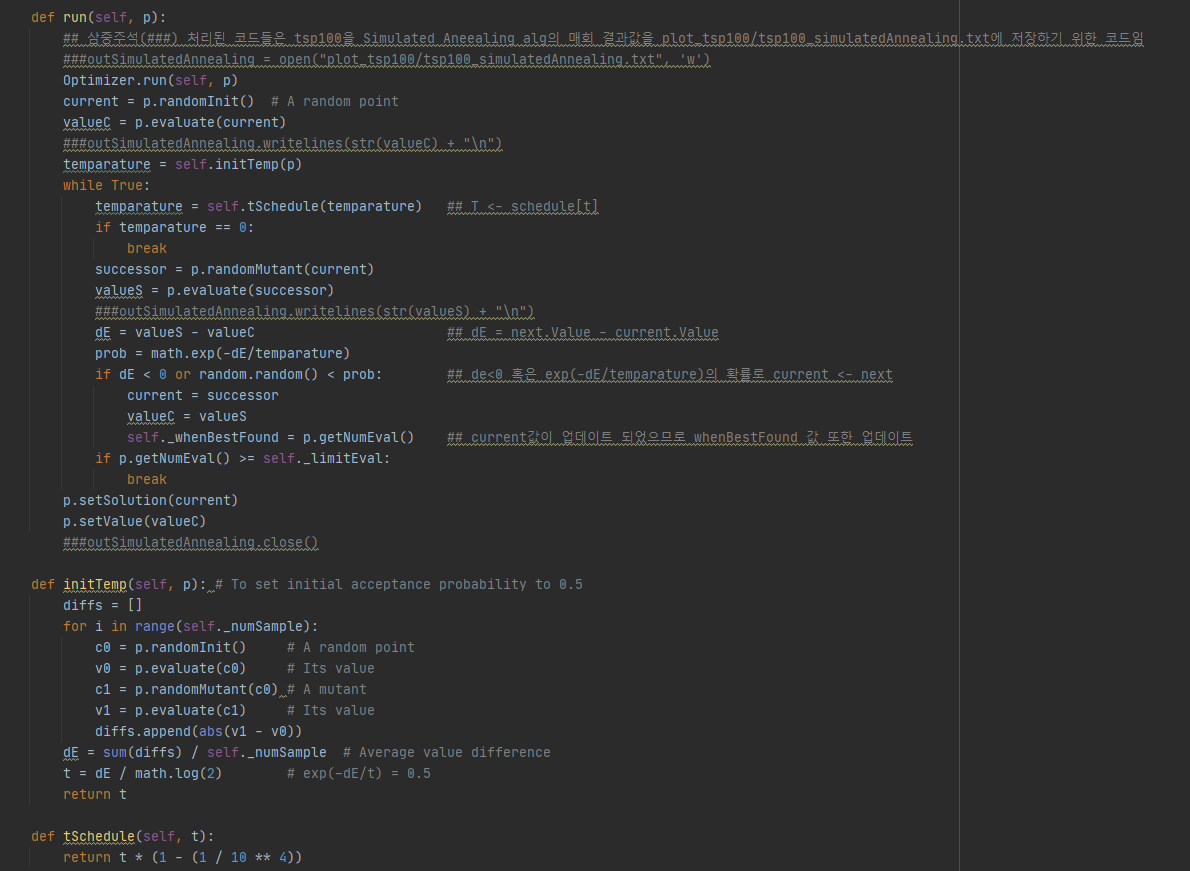
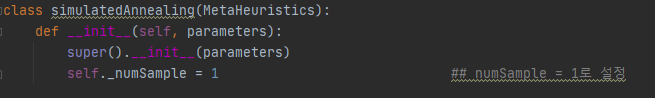


* + 1. 주요 고려 사항:
       1. New user interface에 요구되는 멤버 변수들 중 numRestart, limitStuck를 Optimizer 클래스에 선언, 이를 생성자에서 parameters를 매개 변수로 받아 초기화.
  1. MetaHeuristics(Optimizer) class
     1. Class 개요:

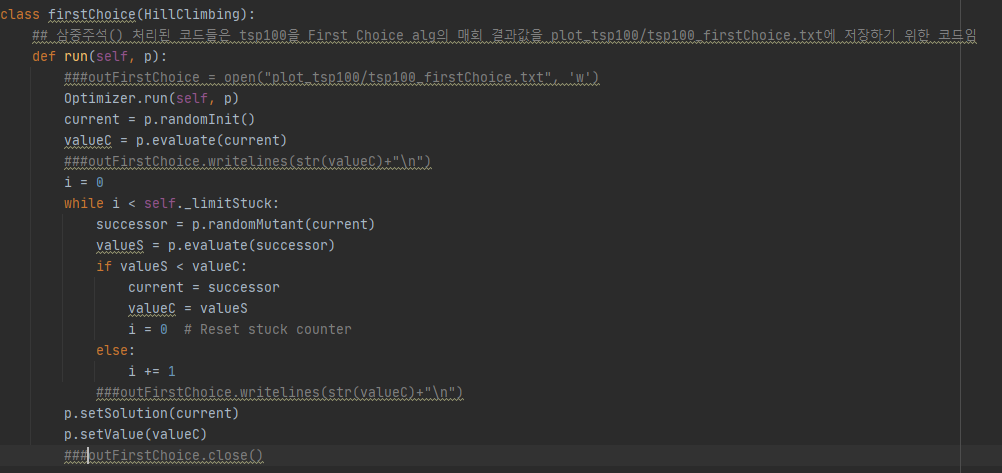
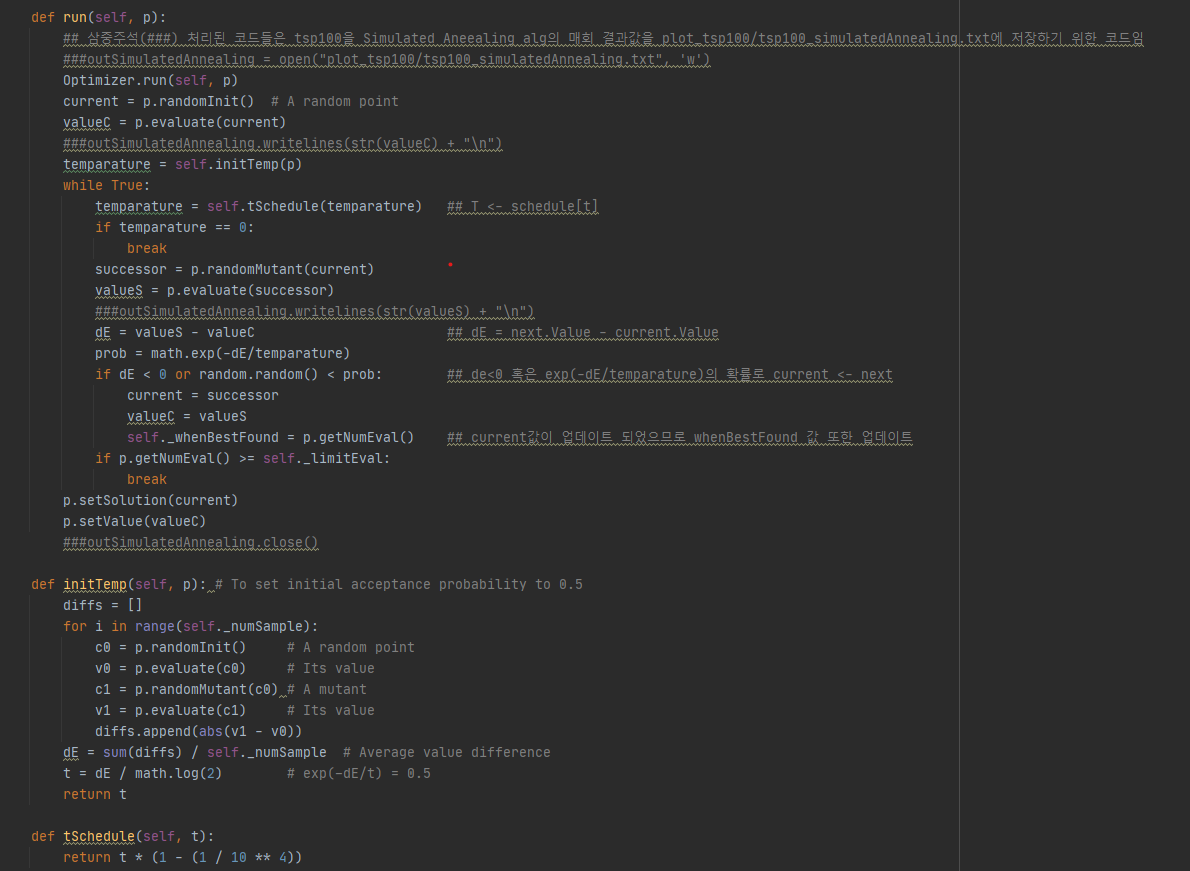
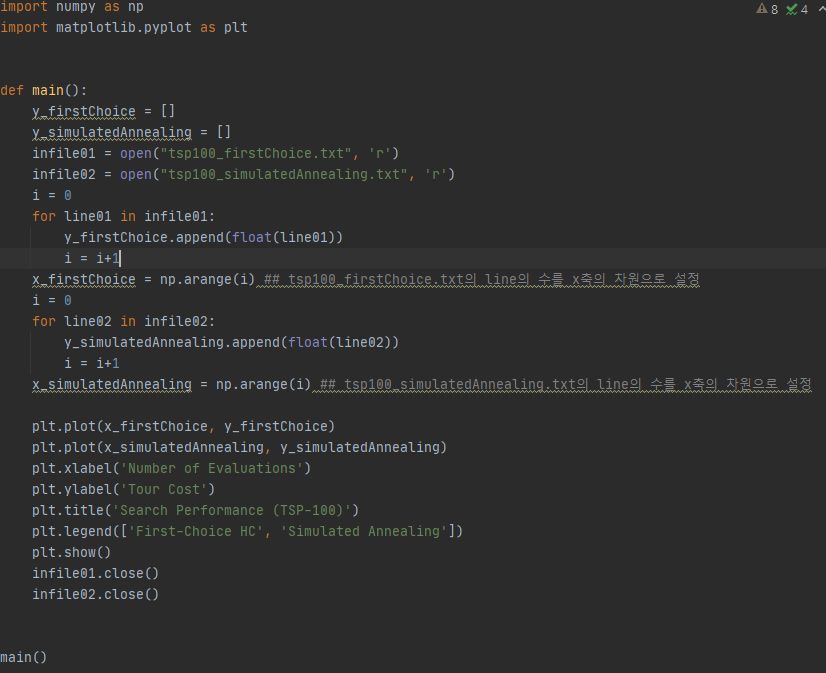
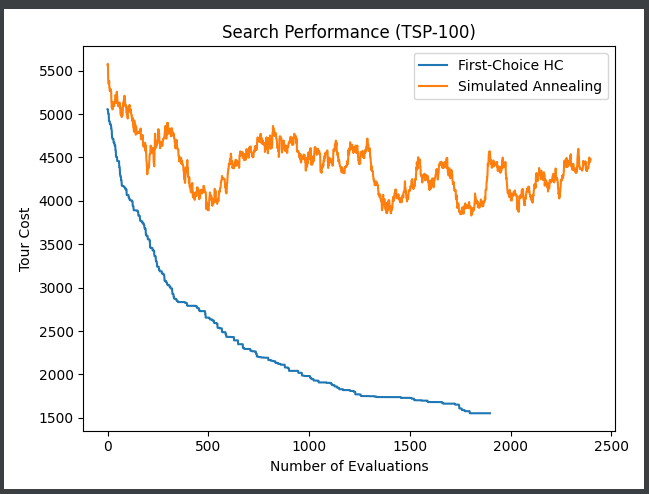
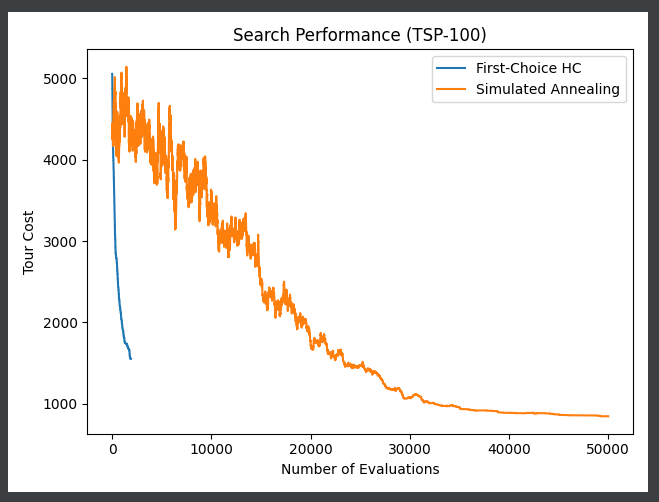


* + 1. 주요 고려 사항:
       1. New user interface에 요구되는 멤버 변수들 중 limitEval를 Optimizer 클래스에 선언, 이를 생성자에서 parameters를 매개 변수로 받아 초기화, whenBestFound를 선언 및 초기화.

Some Useful Codes for Implementing Stochastic Hill Climbing and Simulated Annealing

1. Stocastic(HillClimbing).py
   1. 함수 개요:
      1. Class 개요:
      2. 주요 고려 사항:
         1. Def stochasticBest(self, neighbors, p):
            1. 사전에 제공된 코드를 사용. neighbors를 매개변수로 받아 그중 확률에 따라 임의의 neighbor를 successor로 지정. 지정할 확률은 각 neighbor의 value 값에 비례.
         2. Def run(self, p):
            1. NumEval = 0으로 초기화 후 랜덤한 current와 current의 값을 생성및 초기화.
            2. current의 neighbors를 생성 후 stochasticBest(self, neighbors, p)로 succesor를 특정한다. 이후 current의 value와 비교해 successor의 value가 작을 경우 current를 successor로 업데이트 한다.
            3. 그렇지 않을 경우 iterator가 증가하며 iterator가 limitStuck에 도달했을 때 업데이트를 멈춘 후 current와 current의 value를 p에 저장한다.
2. SimulatedAnnealing(MetaHeuristics) class
   1. Class 개요:
   2. 주요 고려 사항:
      1. Def \_\_init\_\_(self, parameters):
         1. 매개변수 numSample을 선언 및 1로 초기화. (Def initTemp(self, p)에서 사용)
      2. Def initTemp(self, p):
         1. 사전에 제공된 코드를 사용. 초기 온도 값을 지정.
      3. Def tSchedule(self, t):
         1. 사전에 제공된 코드를 사용. 매 시행 당 온도 값의 변화시킴.
      4. Def run(self, p):
         1. NumEval = 0으로 초기화 후 랜덤한 current와 current의 값, 초기 온도값을 생성및 초기화.
         2. 업데이트 시작. Def tSchedule(temparature)로 temparature 값 조정. temparature ==0 일 시 업데이트 종료.
         3. current의 randomMutant를 successor로 지정한 뒤 그 값을 생성 및 초기화.
         4. Successor의 값과 current의 값의 차를 dE로 지정, exp(-de/temparature)를 업데이트 확률로 지정.
         5. dE가 0보다 작거나 그렇지 않을 경우 업데이트 확률로 current를 successor로 업데이트.
         6. NumEval이 limitEval에 도달할 경우 업데이트 종료.
         7. current와 current의 값을 p에 저장.

Plotting the Progress of Search by Using Matplotlib

1. Exp.txt
   1. Algorithm = First-Choice or Simulated Annealing Problem = tsp100, limitStuck = 100, delta = 0.01, numRestart = 1, limitEval = 2400 or 50000, numExp = 1로 지정.
2. FirstChoice(HillClimbing) class
   1. Class 개요:
   2. 주요 고려 사항:
      1. 삼중주석(###) 처리된 라인으로 매회 current의 value를 “tsp100\_firstChoice.txt”에 기록.
3. Simulated Annealing(Metaheuristic) class
   1. Class 개요:
   2. 주요 고려 사항:
      1. 삼중주석(###) 처리된 라인으로 매회 current의 value를 “tsp100\_simulatedAnnealing.txt”에 기록.
      2. 평가 횟수에 따라 두 가지 버전으로 current의 value를 기록. (First-Choice 알고리즘과 유사한 횟수(limitEval = 2400)와 current의 value 값이 평평해지는 지점(LimitEval = 50000))
4. plot\_tsp100/main.py
   1. 함수 개요:
   2. 주요 고려 사항:
      1. First-Choice와 Simulated Annealing 알고리즘의 매회 current value 값을 기록한 두 텍스트 파일로 플로팅. Simulated Annealing의 경우 두 가지 버전으로 기록.
   3. Plot 결과:
      1. Case\_1(SimulatedAnnealing.limitEval = 2400):
      2. Case\_2(SimulatedAnnealing.limitEval = 50000):
   4. 평가:
      1. First-Choice:
         1. Simulated Annealing에 비해 상대적으로 빠른 속도로 best value를 찾아냄.
         2. 그러나 두 알고리즘 모두 best value에 수렴할 때까지 진행한다고 가정할 때 best value의 질은 Simulated Annealing에 비해 상대적으로 우수하지 못함.
      2. Simulated Annealing:
         1. First-Choice에 비해 상대적으로 느린 속도로 best value를 찾아냄. First-Choice 알고리즘이 best value에 도달했을 때 Simulated Annealing 알고리즘은 best value 값이 초기 값에서 크게 벗어나지 못한 것을 확인할 수 있음.
         2. 그러나 두 알고리즘 모두 best value에 수렴할 때까지 진행한다고 가정할 때 best value의 질은 First-Choice에 비해 상대적으로 우수함.