Programming Assignment 07
Search Tool (Version 5)

AI Programming

Overview

In this assignment, you complete the development of your search tool by adding genetic algorithm, another well-known metaheuristic algorithm. Then, you experiment with your search tool to solve the six problems given to you using all the implemented algorithms and compare their performances.

Adding GA

You need to add a subclass 'GA' under 'MetaHeuristics' in the 'Optimizer' class hierarchy. The 'GA' class would need the following class variables:

- popSize: population size
- uXp: swap probability for uniform crossover used in numerical optimization
- mrF: multiplication factor to $(1/l)$ for bit-flip mutation used in numerical optimization
- XR: crossover rate for the permutation code for solving TSP
- mR: mutation rate for the permutation code for solving TSP
- pC: crossover probability (uXp or XR)
- pM: mutation probability (mrF or mR)

The method implementing the crossover operation requires two probabilities pC and pM. If the problem to be solved is numerical optimization, then pC = uXp and pM = mrF. Otherwise, pC = XR and pM = mR. You are provided with some codes for the 'GA' class, based on which you need to implement the 'run' method for GA and other methods that need to be called from the 'run' method. You are also provided with the codes for the problem-dependent methods for GA, which should be added to the 'Numeric' and 'Tsp' subclasses under the 'Problem' class. You should first read and understand these codes well before writing the codes for the 'run' method of the 'GA' class, noticing that the 'run' method will make calls to the provided codes for the 'Problem' class. The provided codes assume that the population of genetic algorithm is implemented as a list of individuals, each of which is again implemented as a list of a fitness value and a chromosome.

Experiments

You are asked to solve the three numerical optimization problems and three TSPs using various search algorithms made available in your search tool and compare their performances. For a statistically meaningful comparison, each algorithm should be tested through ten different experiments and their results should be averaged. You are also recommended to randomly restart the hill-climbing algorithms by ten times. Since the performance of a search algorithm is often sensitive to its parameter setting, you need to go through some empirical trial-and-error process to find the best setting before you start the main experiments. Because of the addition of GA to your tool you need to revise the file 'exp.txt' to contain additional setup information necessary.

Your report of experimental results should include the following numbers for each problem with each algorithm:

- Average objective value
- Best objective value found
- Average number of evaluations
- Average iteration of finding the best solution (only for GA and Simulated Annealing)

More importantly, the report should also include the discussions on your findings.