Coopt with Python DEAP Package for NSGA-II

Juli Mueller

December 13, 2016

To run the code: python nsga2_k.py.

Some more info: I'm not going into too much detail of multi-objective optimization. If you want to know more, let me know. This here is just the bare minimum. Let me know if something is incoherent.
Formulate the problem as a bi-objective problem:

$$\min_{\mathbf{x}} [f_1(\mathbf{x}), f_2(\mathbf{x})]^T \tag{1a}$$

$$\text{such that } \sum_{i=1}^{d} x_i = 1 \tag{1b}$$

$$x_i \in [0,1], i = 1, \ldots, d, \tag{1c}$$

$$f_1(\mathbf{x}) = -\text{Merit}(\mathbf{x}) \tag{1d}$$

$$f_2(\mathbf{x}) = \text{Cost}(\mathbf{x}) \tag{1e}$$

where $\mathbf{x} = [x_1, \ldots, x_d]^T$ denotes the parameters to optimize, $d = 17$ is the number of parameters. Objective (1d) minimizes the negative of the merit functions, i.e., we maximize the merit function. The merit function mtf_mmf is defined in merit_functions.py. I changed this function only by adding K as an input argument (previously hard coded in here).

The objective functions are cheap to evaluate, so I used a multi-objective genetic algorithm to solve the problem. In particular, I downloaded the DEAP python package `https://pypi.python.org/pypi/deap` and adapted whatever I found in the online documentation to the code you gave me. I used the NSGA-II [1] option for evolving the population (consisting of individuals) over the generations (these are the iterations). Initially, we uniformly select points $\mathbf{y} = [y_1, \ldots, y_d]^T$ from the whole parameter space and create the individuals after enforcing (1b) which is done by dividing each point's parameters by the sum over the parameters, i.e.,

$$x_i = \frac{y_i}{\sum_{i=1}^{d} y_i} \ \forall i, \tag{2}$$

see uniform lines 116ff.

Then the fitness function is evaluated (eval_mo, merit and cost, lines 78ff) for each individual. We choose the best individuals of the current generation and mate and mutate them to generate offspring, i.e., the individuals for the next generation ("the fittest individuals survive", good properties of parents are inherited by their children). When doing mutation and mating, we have to enforce (1b), which is done as in (2), see scale lines 122ff. We iterate over all generations and keep the non-dominated solutions (the ones on the Pareto front that is plotted to a pdf-file after the algorithm finishes).

Algorithms parameters are (lines 200-202):

- NGEN: the number of generations
- MU: the number of individuals in each generation
- CXPB: the cross-over probability (needed for generating offspring)

Thus, we do a total of NGEN $\times$ MU function evaluations. Changing the parameter values may influence the goodness of your results, but you can play with this. In the main function, KVEC (line 265) defines all the K values that you want to run. Right now they are all being run, so out-comment or delete whatever you don't want.

References

[1] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving from Nature*, pages 849–858, 2000.