A Guide to Using Evolutionary Algorithms

Shiu Yin YUEN

Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China
Updated: 10 July 2015

Motivation

I found that there are some beginning students who misuse evolutionary algorithms (EAs). This short essay is intended to clarify when EAs should be used and when they should not.

What are Evolutionary Algorithms?

Evolutionary algorithms are the followings:

1. Trial and error methods – Think of random search. EAs are at best methods that are better than random search on average for real world problems. Note that this is a belief only. There is no proof.
2. Stochastic methods – Like random search, the performance of each run is different. Thus it is typically applied to design problems. The algorithm is run a few times and the best solution found is taken. It is not a very good idea to apply it to online control problems, unless you can prove that the algorithm has a small variance. No such proofs exist to my knowledge.
3. Optimization methods – they are used to find better solutions to an existing problem. You cannot use it if you do not have a problem, and need to create a problem.
4. Methods that works when the gradient cannot be computed – so called "derivative free".
5. Methods that works when there is more than one optimum.
6. Methods that cannot give you the global optimum in general. So you should justify its use – e.g. the problem is NP hard and you are content to find a "good enough" solution. This is typically a solution that improves the best known existing solution to your problem.
7. A subset of derivative free methods – EAs, because of the term "evolution", tries to design algorithms by using (perceived) optimization mechanism in nature as metaphors. Note that they are not the only derivative free methods available.

When and When Not to Use Evolutionary Algorithm

If the problem has been formulated by mathematical formulae and mathematical constraints, the so called open box problems, you should not try EAs right away. You should consult the vast literature on operational research (OR) and use techniques such as linear programming, quadratic programming, integer programming, etc. If you use EA on a problem that can be solved (i.e., to find the global optimum) using linear programming, you guarantee that the EA will be inferior to linear programming.

Sometimes you have open box problems, but the mathematics is too complicated such that no existing mathematical techniques can solve it, then EAs can be used in this way: hypothesize the value of a (as small as possible) subset of variables, such that when these values are known, the problem reduces to a conventional OR problem. Then use EA to solve for the values of the variables in the subset. For each instantiation of the variables, apply the OR solution method once. This is called Memetic Computing.

When you have a problem which is a black box problem, then you have to use EA or other derivative free optimization methods. A black box problem is one for which the function form is unknown, and one can only sample the values at a sampling location one at a time, and the function will return the function value. A good example is a variable in a simulation software or a variable measured in a real industrial application.

Many (but not all) EAs require the application users to provide the ranges of the optimization variables. This can be a problem if the user has no idea what is the range, e.g. when $-\infty \leq x \leq \infty$.

If sampling the black box will change the black box, then you have to use dynamic EAs.

If your problems have multiple objectives (MO), you have to use a MOEA.

Sometimes, you have a problem class (e.g. Travelling Salesman problems (TSP)) and you apply an EA to solve it. There are usually some parameters specified by the EA inventor. Don't change the settings of these parameters. If you change them, you are inadvertently tuning the EA to fit the problem instances. If you have a problem class, it is not enough just to show that the EA works well on a few instances of the class. At the least, you need to show that the EA works well on all representative instances of the class. Sometimes, researchers make do with showing that the EA works well on all instances of a well known benchmark suite (e.g. TSPLIB). If so, you are making the implicit assumption that the suite is representative of all TSP problems. Please make this assumption explicit when you write a paper to report your result – as it is often highly suspicious if one makes a claim that a benchmarking suite is representative of the problem class, and it is highly likely that there is no proof supporting the claim.

Rarely, you have a single problem with some unknown parameters. The solution to every point in the search space is completely defined but exhaustive search is impossible because it is huge. This may occur in some mathematical or scientific problems. For example, find the building layout that maximizes the airflow in a district in Hong Kong. Then the problem is to get the parameters that give the best possible. In this case, by all means use an EA or any other method. Any set of parameters that give a better result is in this case an advance in science.

Extended Readings

K. Sörensen, “Metaheuristics – the metaphor exposed,” Intl. Trans. in Op. Res., vol. 22, no. 1, pp. 3-18, 2015.

Z. Michalewicz, “Quo vadis, evolutionary computation? On a growing gap between theory and practice,” *Lecture Notes in Computer Science*, vol. 7311, pp. 98-121, 2012.

L.M. Rios and N.V. Sahinidis, “Derivative-free optimization: a review of algorithms and comparison of software implementations,” *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247-1293, 2013.