A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms

Evolutionary algorithms (EAs) are widely used for solving multiobjective optimization problems (MOPs) because of their ability to find the solution in many different environments and with many types of variables. However, problem with EAs is that they usually require high number of objective- and/or constraint function evaluations which may in some cases be either very time consuming or computationally expensive. This introduces the need for creating approximations of the functions. Sometimes these approximations may be called “metamodels” or “surrogates” where computational model is created to represent the objective function.

Approximation can be applied to multiobjective optimization problem in different ways. Three different ways are problem, function and fitness approximation. Problem approximation means replacing the problem with one that is computationally easier to solve. Function approximation means replacing computationally expensive functions with the simplified ones which are faster to evaluate. Fitness approximation means that instead of getting individuals fitness value by evaluating the fitness function, fitness value is derived from existing near-by individuals. Most popular from these three is the function approximation.

When using function approximation, generally the workflow is divided to two stages. The first stage uses general EA for predefined number of generations and results are stored for building the metamodel. After the first stage, metamodel is built for each objective, and constraint function which are computationally too expensive. In stage two EA with these metamodel(s) are used. Individuals created by the stage two can be selected for re-evaluation with the original objective or constraint functions and results from the re-evaluation can be stored and used for improving the metamodel(s). There exist different options for using metamodels with MOPs, for example, one metamodel can be used to represent all the objective functions or different metamodels can be created for each objective. Also, methods like scalarizing the MOP into single objective problem and using metamodel for solving the scalarized problem are possible.

Algorithms that are based on single metamodel for all the objective- and constraint functions can be roughly divided into two categories. Kriging based algorithms that use Kriging or Gaussian process regression for creating the metamodel. Non-kriging-based algorithms use methods like neural networks and machine learning to create the metamodels. Currently Kriging based algorithms are more widely used than non-kriging-based ones.

When selecting an algorithm for the real-world use, two key metrics should be considered, maximum number of function evaluations allowed and problem dimensions in objective and in decision space. If algorithm requires too many function evaluations, it may be infeasible in real-world use case. Also, high number of objectives or decision variables may limit suitable algorithms since most of the current algorithms are tested with only three or two objectives and less than 30 decision variables.