Block Coordinate Descent

Max Iterations = Roof of (attributes / n); where n is the number of attributes in 1 block

X0 = Starting Pt

Var\_order = Order for Iterations

While Epochs < Max Epochs:

Do {While Iterations < Max Iterations:

Do {x = Sampling(n, method); Sampling: Creates ‘n’ shaped data based on method

X\_data = Sampling\_fn(X0, x, Iterations);

[Sampling\_fn: Based on x and Iterations creates X\_data which has constant values for attributes not in that Var\_order]

Y\_data = execute\_C\_file(X\_sdata)

model = Alamo\_train(X\_data, Y\_data)

X\_min, Y\_min = scipy.optimize(model)

Store X\_min, Y\_min in excel file

X0 = X\_min}

Requirements:

In this project, we want to implement a hybrid machine learning and optimization approach to solve black-box optimization problems. Black-box problems are problems for which we do not have an algebraic formulation of the problem we want to optimize. Instead, we are relying on a simulator or experiment to measure the objective function at a specific point in the variable space. For instance, we could use Aspen to optimize a chemical process or we could use ARENA to optimize traffic flow. In the former case, we would provide temperatures and flowrates to Aspen and Aspen would return the conversion rate of the process to be maximized.  In the latter case, we could specify the number of cars through specific routes and ARENA would return traffic times that we would want to minimize. The attached paper by Rios & Sahinidis reviews this area of research from the point of view of applications and algorithms. This paper compared a large number of algorithms, called derivative-free optimization algorithms, that have been devised for this class of problems. The primary conclusion was that these algorithms fail to provide good solutions for problems with more than about a dozen degrees of freedom.

 This project will investigate the following class of algorithms:

1.     Call the simulator a number of times and build a surrogate algebraic model of the black box.

2.      Apply an optimization algorithm to the surrogate model.

3.      If some termination condition is met, terminate; otherwise, repeat the first two steps.

Students in this project will work on different approaches for the above steps.  In terms of the types of models that we build in Step 1, we will consider:

A.          Building global models (over the entire domain of interest)

B.          Building local models (over trust regions)

In terms of how we build models in Step 1, we will consider the following machine learning algorithms:

a.           ALAMO. Download the latest ALAMO from <https://www.minlp.com/download> and get familiar with it (look also into the manual).

b.           Lasso and other alternative machine learning techniques, for example these in scikit-learn (<https://scikit-learn.org/stable/>).

In Step 2 of the algorithm, we will rely on the BARON optimizer that is available through PYOMO. Make sure that your Python installation includes PYOMO. Then, install

BARON from <https://minlp.com/baron-downloads> (look also into the manual).

Finally, in Step 3 of the algorithm, we will calculate the difference between the surrogate model and the simulation at the solution obtained in Step 2. If this difference is sufficiently small, we will terminate. We will also terminate if we have exceeded a pre-specified number of iterations.

To begin with this project, install Python, Pyomo, ALAMO and BARON on your laptop. Then, think in terms of writing a python code that implements the above main algorithmic loop. Later, we will provide you with more details on how to proceed. Those of you who work on approach A (global models) will look into implementing an initial design of experiments to sample the domain. Those of you who work on approach B (local models) will implement a trust region algorithm. Those of you who use ALAMO for the learning will rely on ALAMOPY. Those of you who work with other machine learning tools will rely on scikit (or any other package of your choice). Finally, all of you will use a collection of 500 test problems discussed in the Rios & Sahinidis paper, for which we know the optimal solutions. These problems will be used for training and testing of your algorithms. Once we are past these test problems, we will consider applications, including optimization of water-ethanol systems, heat exchanger design, and others. More details about these steps will be provided later in the semester. For the time, assume that your black box simulates the function x^2 and use this function to write a simple code implementing the above algorithm.