Integrated Machine Learning and Optimization in Python

Master of Science Degree Research Project

December 2019

Swapnil Agrawal

Advisor - Nicholas V. Sahinidis

Department of Chemical Engineering

Contents

Chapter 1

Abstract

As the system are becoming more complex, it is easier to experiment them rather than understanding. Hence, solving black-box function optimization problem is becoming increasingly important. This project aims to perform black box optimization by integrating machine learning and optimization methods. Three different algorithm are selected and their performance over a black-box function optimization problem is discussed. The best performing algorithm is used to solve a set of 500 black box problems which contains convex as well as non-convex problems. Practical scalibility of these methods with respect to dimension of the search space and non-linearity of the function is tested and discussed.

Chapter 2

Introduction

The objective of this project is to perform black box optimization using machine learning and optimization methods. Black box refers to a system which is so complex that it is easy to experiment rather than understanding it. In some setups, the function is not at all available in the algebraic form though we have access to function output on providing it some input.

Now a days, as the systems are becoming more complex, black box optimization is becoming increasingly important. Black-box optimization has many applications, for example, optimizing number of reactors in a simulation or calculating hyper-parameter for a machine learning algorithm such as depth in a neural network etc.

Black-box optimization is the task of optimizing a black-box function with limited number of function evaluation. Evaluating the function can be expensive in terms of money or time. Our aim in this project is to find an algorithm which can find the function optima using as less number of function evaluations as possible.

In this report, we discuss about a python interface which integrates machine learning techniques, ALAMO and BARON to develop a surrogate model of the black-box function and then optimize it. The input to this program is a black box function, the bounds on variables, the limit on function evaluation and a stopping criteria. With all these as inputs, the system returns the calculated optimum value of the function within the specified number of function evaluations. The performance of the algorithm is measured by solving a set of 500 black-box problems and comparing the results with the known optimum value. Finally, the algorithm is tested for a aspen black-box function for which variables are Temperature, molar flow rate and pressure and the objective is to minimize the energy of the operation. These results are analyzed and the performance of the algorithm and the possible improvements are discussed in the later sections.

Chapter 3

Literature Survey

Due to the vast applications of black-box optimization, it has been studied in many papers. The approach to black-box optimization can be mainly classified into two categories, indirect and direct optimization. Direct approach, also known as derivative free approach uses only function evaluation and no derivative information to calculate optima. Whereas, indirect approach uses surrogate model or function derivative for the calculation. [1]

Previous research have approached black-box optimization using metaheuristic methods such as Particle Swarm Optimization (PSO) [2], Ant Colony Optimization etc. These approaches usually require more evaluations of black-box function and are thus expensive methods [6].

The other approach involve Bayesian Optimization which is considered a powerful tool for optimizing non-convex functions. In terms of required number of objective function evaluations, Bayesian optimization methods are considered to be some of the most efficient techniques ([3], [4], [5]) for black-box problems of low effective dimensionality.

In this project, we aim to solve convex as well as non-convex black-box function and to explore practical scalability of the methods with respect to the dimension of the search space and nonlinearity.

We have tested 3 different algorithm on a black box function camel 6. It is a two-dimensional test problem, referred to as the six-hump camel back function, exhibits six local minima, two of which are global minima at [0.0898, 0.7126] and [0.0898, 0.7126] [1]. These algorithm uses Coordinate Descent and Trust Region methods. The algorithms are described in the next chapters in detail.

Trust Region Method

Trust-region method works in a way that first define a region around the current best solution, in which a certain model (usually a quadratic model) can to some extent approximate the original objective function. It then take a step forward according to the model depicts within the region. The size of the trust region in each iteration would depend on the improvement previously made.

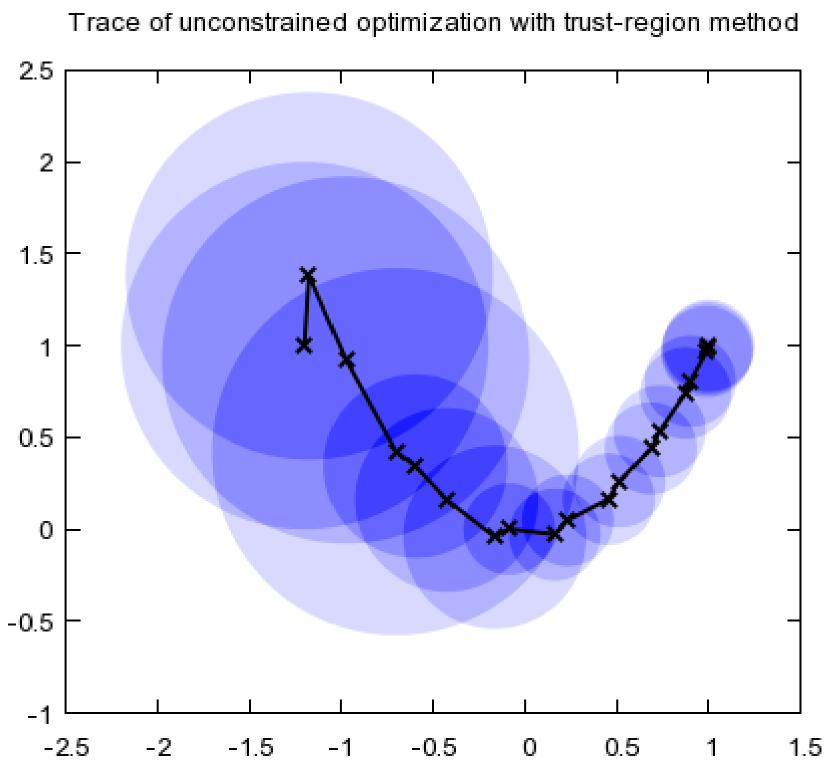


Figure 3.1: Progress with Trust Region Method in 2-dimension

Coordinate Descent Method

It is an optimization algorithm that successively minimizes along coordinate directions to find the minimum of a function. At each iteration, the algorithm determines a coordinate or coordinate block, then exactly or inexactly minimizes over the corresponding coordinate hyper plane while fixing all other coordinates or coordinate blocks.[7]

descent.png descent.png

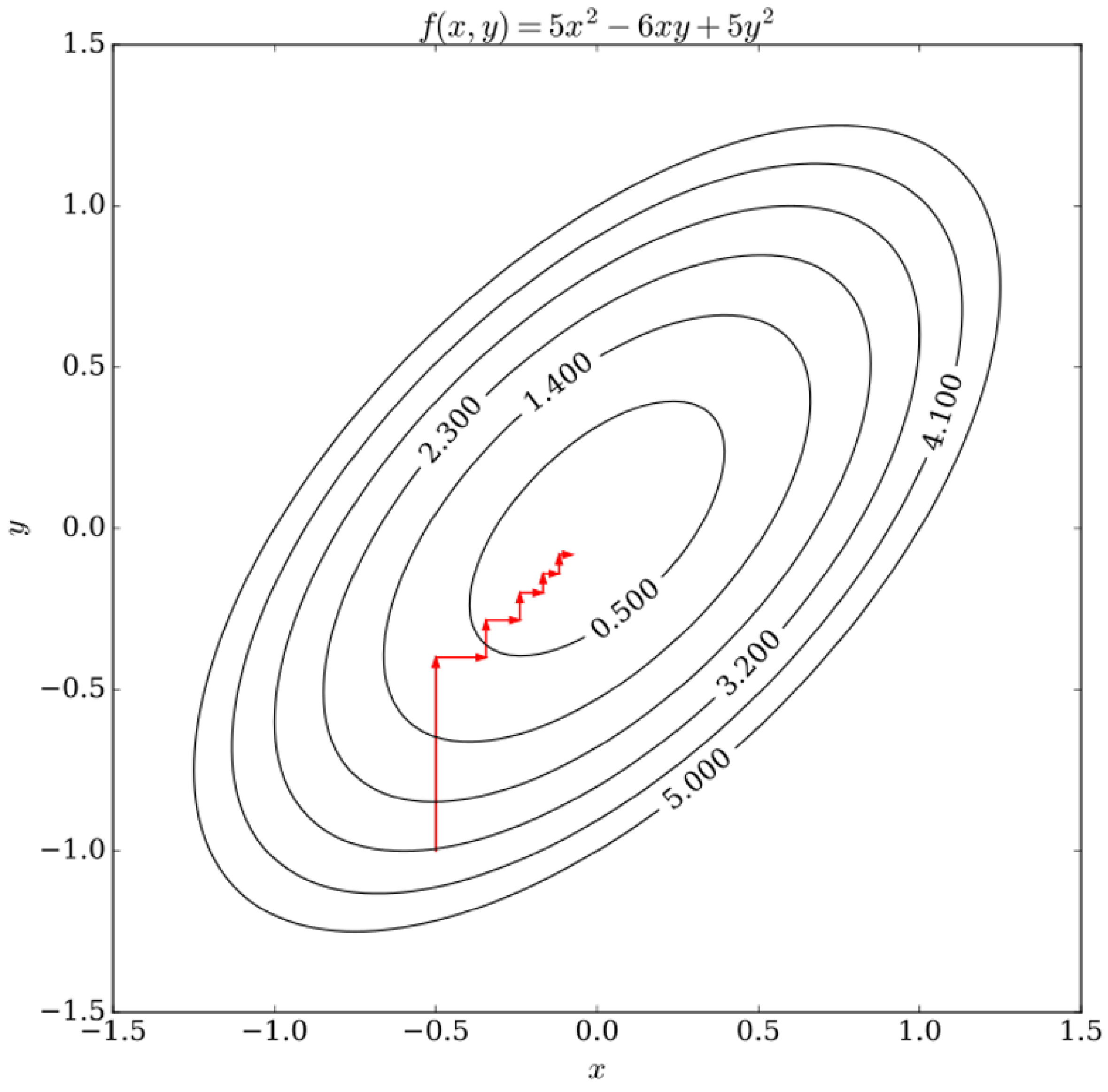


Figure 3.2: Coordinate Descent Method progress in 2-dimension

Chapter 4

Proposed Algorithm

For solving black-box optimization problem, our general approach is, select a region, sample few points in that region using a sampling method which are described later in this section. These points can then be fitted to an algebraic function using machine learning models or ALAMO. The fitted model is optimized using baron. The process is repeated until we have reached a stopping criteria or have achieved convergence.

Using different combination of sampling methods, sampling region and the update rules can give us different final results. In this project, I experimented with five different sampling methods and three different methods for region selection, namely global optimization method, trust-region method and cyclic coordinate search method. For developing surrogate model, ALAMO and Sci-kit learn packages are used. The section discusses these sampling methods and algorithms in detail.

4.1 Sampling methods

To develop a surrogate model which is a good approximation of the actual black-box function it is important to cover the input space properly. Five variants of sampling methods are used and their performance on a black-box function is compared later in the result section.

1. Random Sampling : In this sampling method, every point is chosen randomly and entirely by chance. Each point on the sampling space has an equal probability of getting selected. An example of random sampling in 2 dimension is shown in figure below.
For sampling large number of points, random sampling is a good method, but using it for less number of points may result in cluster formation and missing an area which might contain important information about the function.

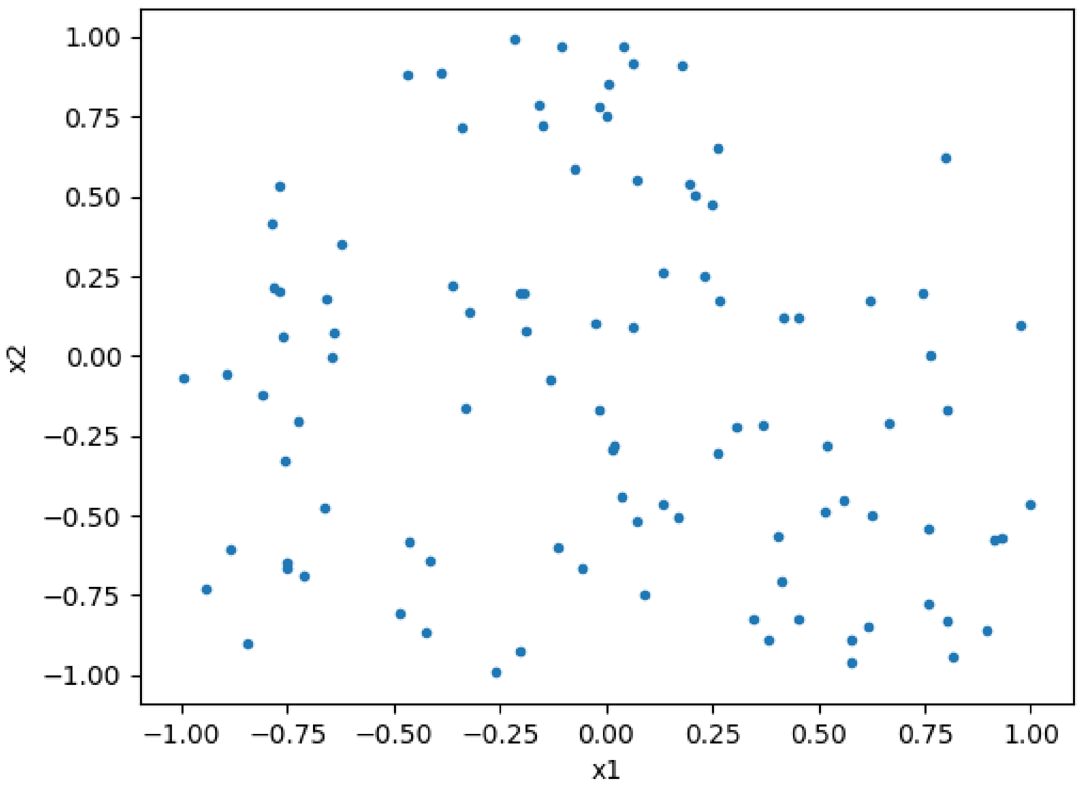


Figure 4.1: Random Sampling in 2 Dimension

2. Sobol sequence : Sobol is a quasirandom sequence, i.e. less random than the random sampling. There is some relation between different sample points. These are good for convergence as these sequences tend to sample space more uniformly than random numbers. The figure below shows sobol sequence for a sampling size of 100 points in 1 dimension.

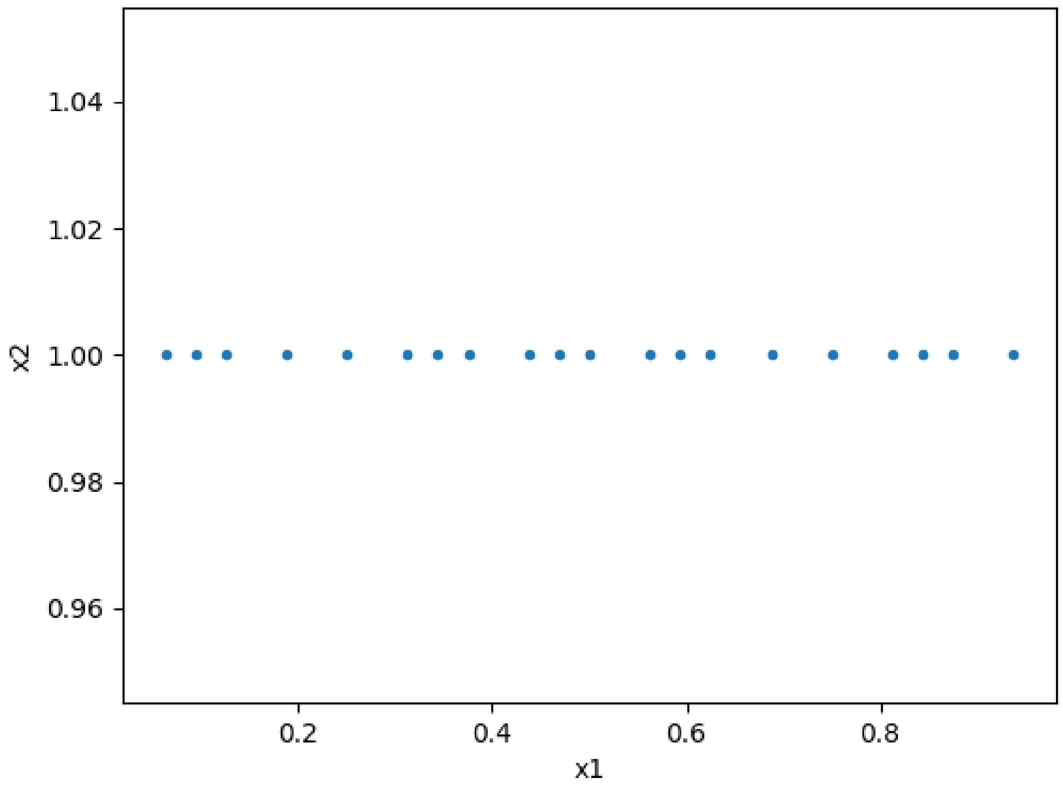


Figure 4.2: Sobol Sampling in 2 Dimension

3. Latin hyper-cube : It generates a near-random sample in a multidimensional distribution. In 2 dimensional, it will be a square such that there is only one sample in each row and each column. Generalization of this concept to an arbitrary number of dimensions, gives us a sample where there is only one point in each axis-aligned hyper-plane containing it.

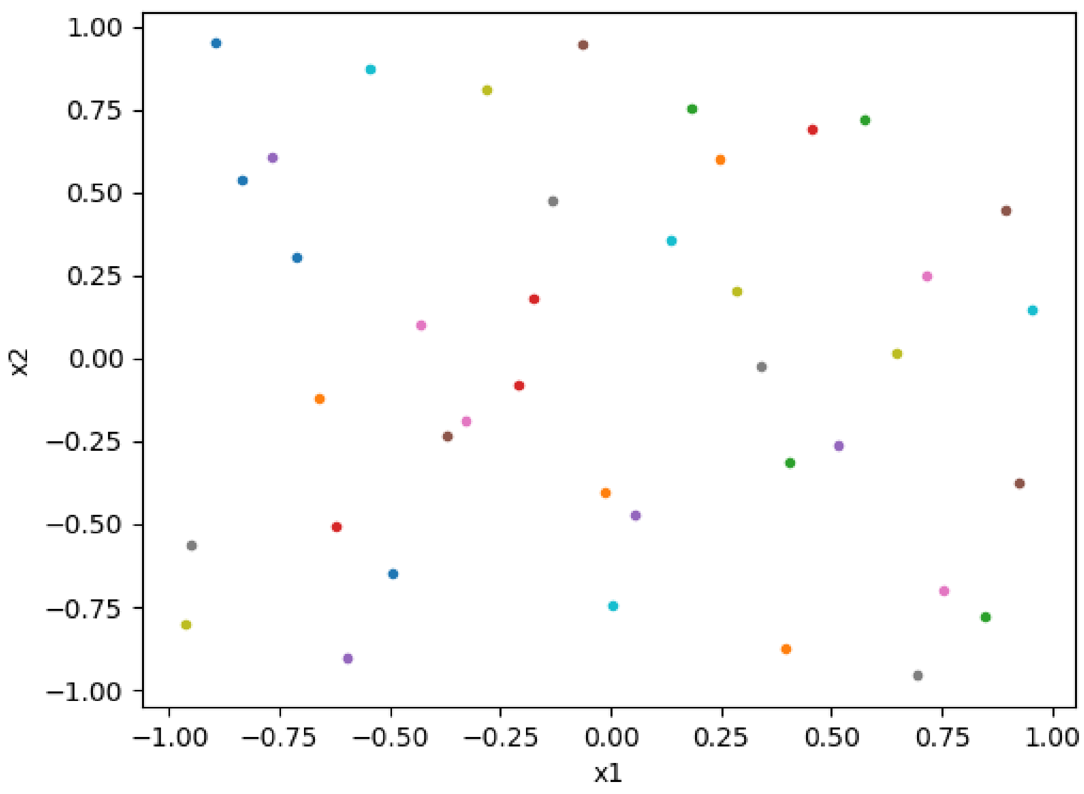


Figure 4.3: Latin Hypercube Sampling in 2 Dimension

4. Hammersley : Hmmeresley is also a quasirandom sequence same as sobol except that here the vander Corputs sequence is generalized to higher dimension to generate sample.

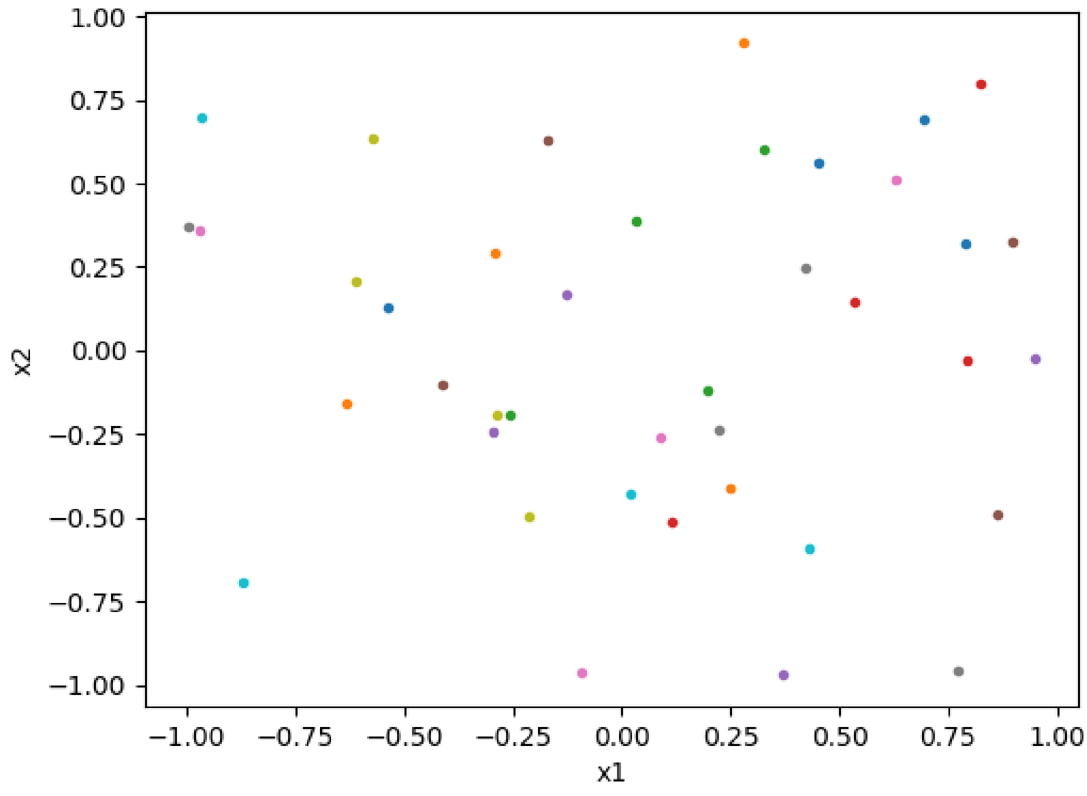


Figure 4.4: Hammersley Sampling in 2 Dimension

5. Grid Sampling : The input space is divided into a grid, the size of which should be equal to the sampling size. Then a point from each grid can be chosen either randomly or the mid of the grid. Grid search ensures that the whole sample space is covered while sampling. Though in this case, the variations in the function might not be covered properly as there can be cases where all grids are not equally informative.

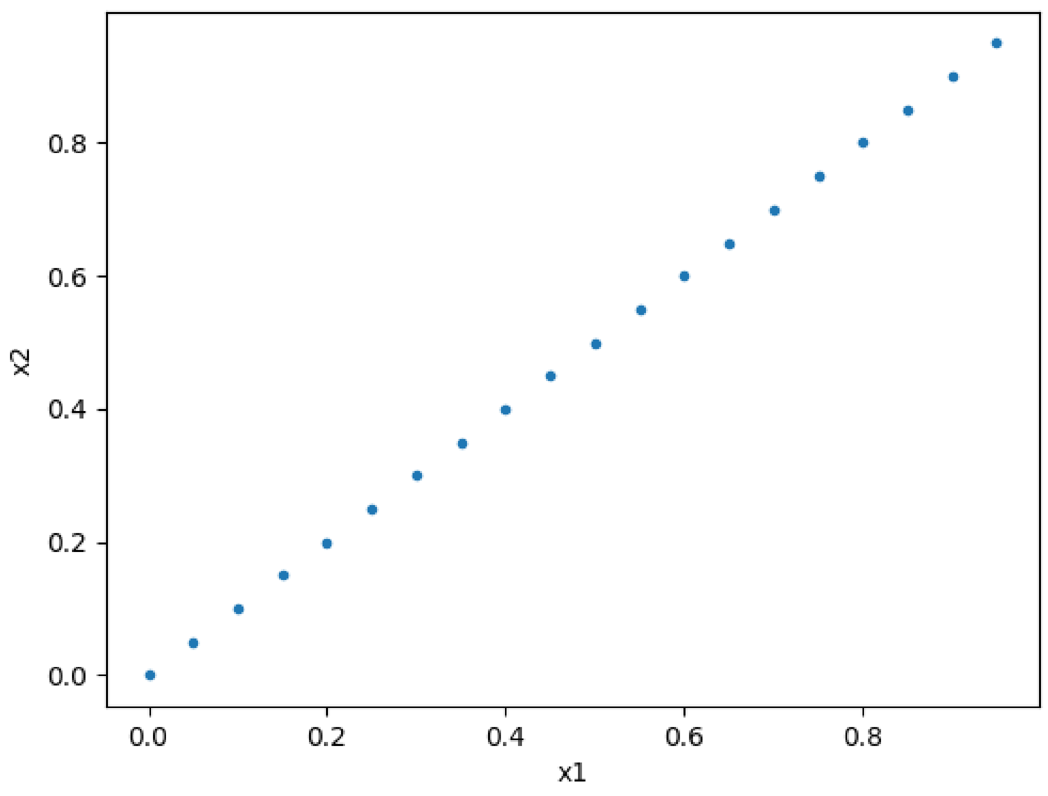


Figure 4.5: Grid Sampling in 2 Dimension

4.2 Building Surrogate Model

1. ALAMO: It is a software which is used for building nearly accurate algebraic model given the data. A python interface, alamoPy, is available to use ALAMO functionality with python. Given the sampled input data, alamoPy returns an algebraic function which can then be used for finding optima.

In this project, the sampled input data is fed into alamoPy and the quadratic function that it returns is optimized using BARON or minimize function from python.

2. Sci-kit Packages: Sci-kit learn is an open source machine learning library for implementing machine learning algorithms. Given the dataset, the machine learning algorithm from the package can be used to build a surrogate model. Along with the model parameters, it also provide details about how good the model is.

4.3 Overall Algorithm

1. Trust Region Method

In this approach, a point is selected randomly in the provided sample space and a region around that point is selected. More points are sampled in the region and a surrogate model is built. The model is optimized and the stored optimum is updated if there is an improvement in the value.

For the next iteration, the region is expanded or shrank on the basis of whether there is an improvement or not in the optimum value.

Algorithm 1 Trust Region Method

```
INPUT : Black-Box function (as a c source code file), Input file, output file, Number
of function evaluation, Variable bounds
niter = 0
x0 = Randomly selected starting point
z0 = black-box function value at x0
while(niter < Niter):
- Select a region around x0
- Sample more points in the region
- Compile black-box function and Calculate output for all the sampled points
- Build a surrogate model
- Use BARON to optimize this model
- Update stored optimum value (z0) only if there is an improvement
- if (improvement in optimum value) : shrink region
- else: expand region
- niter = niter + 1
```

2. Global Search Method

The only difference in global search method and trust region method is in the sampling space. In global search method, the overall area is used as a sampling space and it is not changed with iterations.

Algorithm 2 Global Search Method

```
INPUT : Black-Box function (as a c source code file), Input file, output file, Number
of function evaluation, Variables bound, number of points (n)
niter = 0
z0 = 1e9 (a very high number)
while(niter < Niter):
- Sample n number of points
- Compile black-box function and Calculate output for all the sampled points
- Build a surrogate model
- Use BARON to optimize this model
- Update stored optimum value (z0) only if there is an improvement
- niter = niter + 1
```

3. Cyclic Coordinate Search Method

This method uses a combination of global search and trust region method along with coordinate descent method.In each iteration a coordinate or block of coordinates are selected and the optimum is calculated along these dimensions keeping all other coordinates fixed.

The algorithm starts with complete sample space and then in the following iterations, it is decreased or increased based on whether there is an improvement in the optimum value.

Algorithm 3 Cyclic Coordinate Search Method

```
INPUT : Black-Box function (as a c source code file), Input file, Output file, Number
of function evaluation, Variables bound, Niter

niter = 0 z0 = 1e9 (a very high number)

while(niter < n):
- Select a coordinate
- Sample along that coordinate
- Compile black-box function and Calculate output
- Use this surrogate model to find function optima along that direction
- Build a surrogate model
- Use BARON to optimize this model
- Update stored optimum value (z0) only if there is an improvement
- if (improvement in optimum value) : shrink region
- else: expand region
- niter = niter + 1
```

The above algorithm is also described in the following process flow chart.

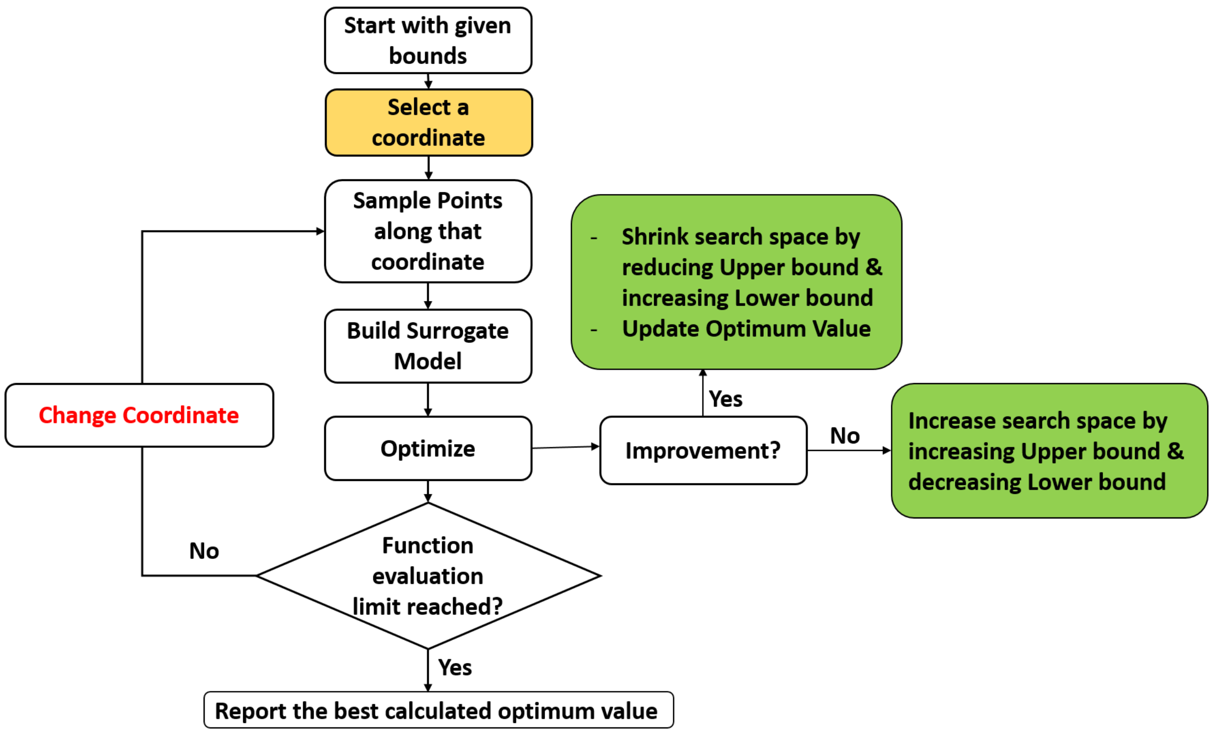

Start with given bounds
Select a coordinate
Sample Points along that coordinate
Build Surrogate Model
Optimize
Improvement?
Yes
- Shrink search space by reducing Upper bound & increasing Lower bound
- Update Optimum Value
No
Increase search space by increasing Upper bound & decreasing Lower bound
Function evaluation limit reached?
No
Change Coordinate
Yes
Report the best calculated optimum value

Figure 4.6: A combination of local search and global search with coordinate descent method

Chapter 5

Results

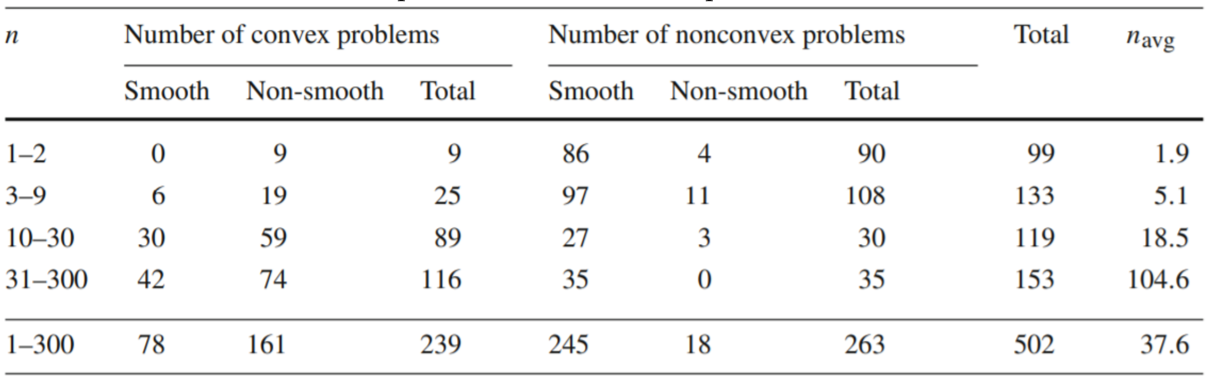
5.1 Experimental Setup

All computations were performed on Intel 2.7 GHz processors running Windows and Python 3.7.2. The 500 black-box problems are solved using cyclic coordinate search method using a limit of 2,500 function evaluations in each run. For all of these problems, number of variables, upper and lower bounds on all the variables and the starting point is provided. For these problems, we also have the optimum value to compare the performance of our algorithm. The various hyper-parameter such as number of sampling points in one iteration, sampling method and order of the surrogate function are optimized by minimizing the difference between known optimum and the calculated optimum within 2,500 function evaluation.
The algorithm is then tested on an aspen problem which had 8 variables and is solved with 20,000 function evaluations. The function ran for 2 CPU days and provided result within 1 percent of the known optimum value.

5.2 Test Problems

The set of 500 black-box problems are used for testing the algorithm and hyper parameter tuning. These problems have a mix of convex, non-convex, smooth and non-smooth problems. A total of 239 of the test problems are convex, while 263 are non-convex. The number of variables (n) ranged from 1 to 300, with an average number of variables (navg) equal to 37.6 [1]. Table and Figure below from Rios et. al. [1] presents the distribution of problems by dimensionality and by problem classes.

of test problems.PNG of test problems.PNG



n	Number of convex problems			Number of nonconvex problems			Total	n_{avg}
	Smooth	Non-smooth	Total	Smooth	Non-smooth	Total		
1–2	0	9	9	86	4	90	99	1.9
3–9	6	19	25	97	11	108	133	5.1
10–30	30	59	89	27	3	30	119	18.5
31–300	42	74	116	35	0	35	153	104.6
1–300	78	161	239	245	18	263	502	37.6

Figure 5.1: Characteristic of test problems

problems details.PNG problems details.PNG

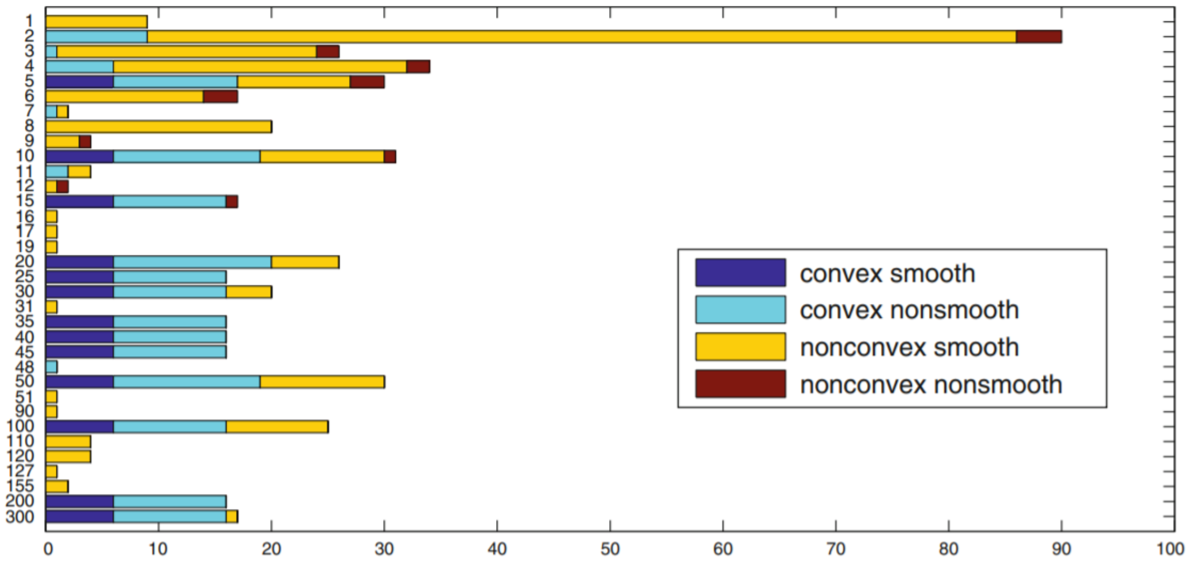


Figure 5.2: Distributions of problems by dimensionality and classes

3. Different Algorithms

5.3 Hyper Parameter Tuning

The parameters to be decided for the given algorithm are sampling method and the number of sampling points that should be used in one iteration. Camel6 blackbox function is tested and the performance is compared by varying these parameters.

1. Varying Sampling Method The dependence of variation of result is observed with variation of sampling method for a black-box function camel6.The result for the different sampling points are compared in the following figure.

method result.png method result.png

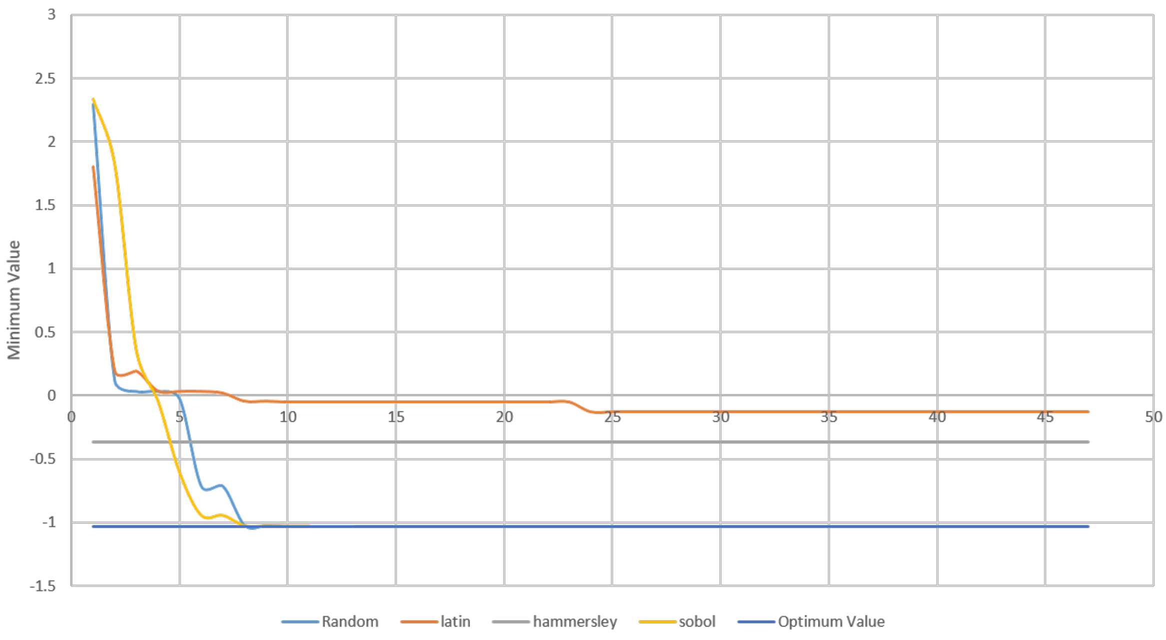


Figure 5.3: Progress of solution with function evaluations for different sampling methods

2. Varying Number of Sampling Points Using sobol sequence as a sampling method, camel6 black-box function is solved with number of sampling points varying from 10 to 80. The results for comparison of time and number of evaluations required to converge are shown in figure below.

of points result1.png of points result1.png

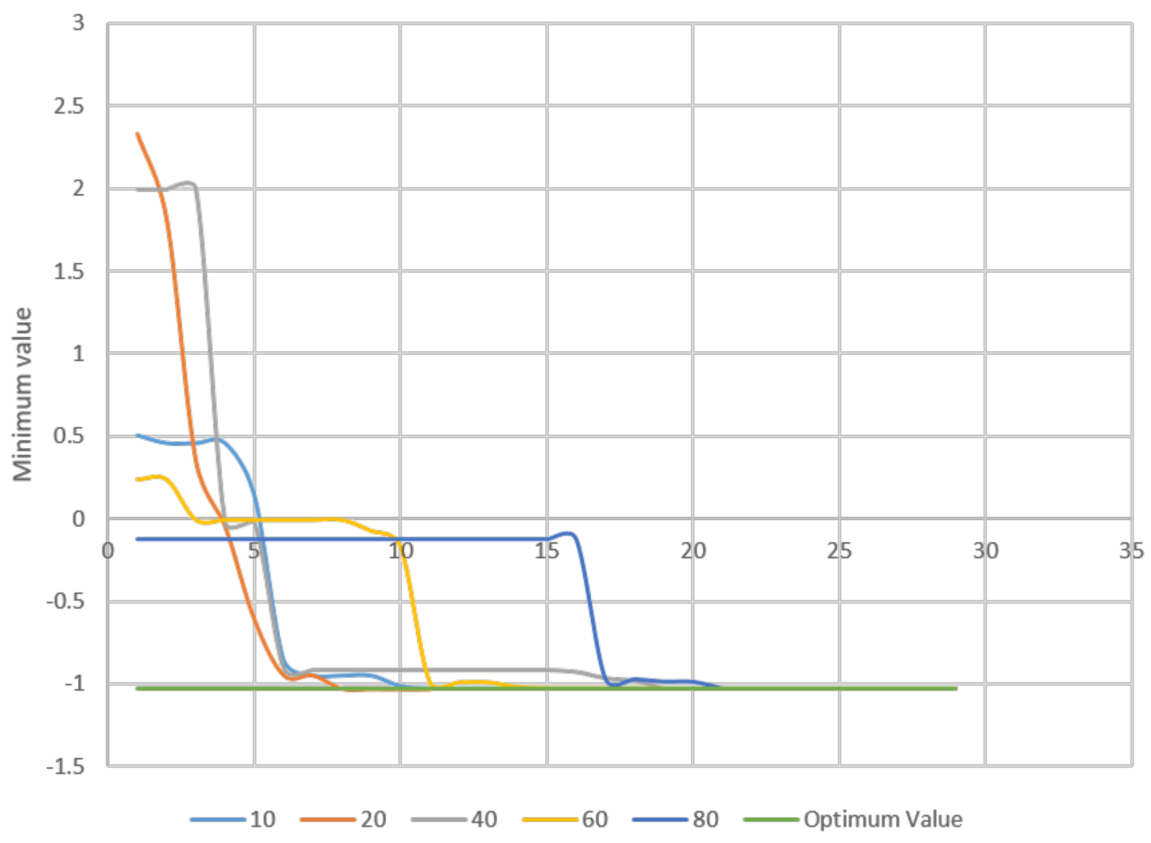


Figure 5.4: Progress of solution with function evaluation for varying number of sampling points

of points result2.png of points result2.png

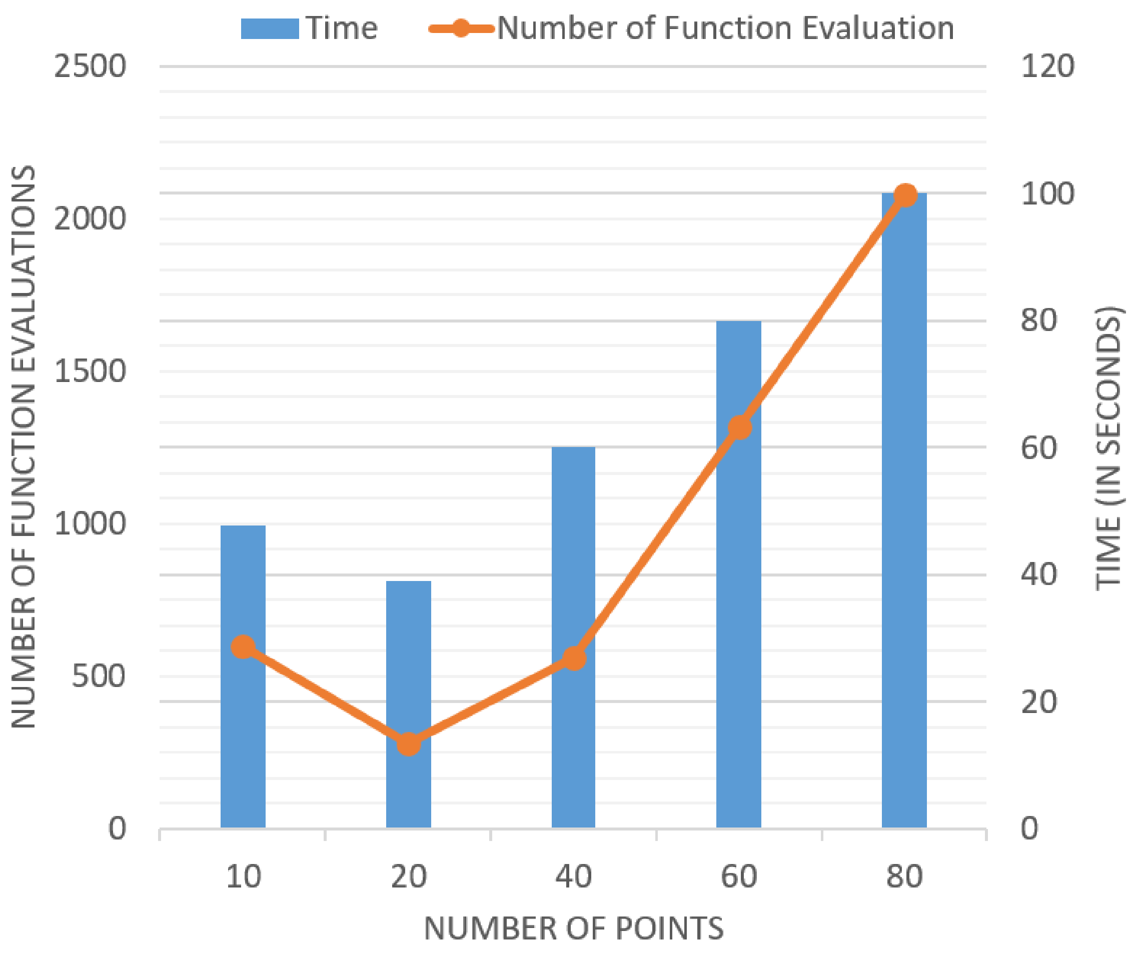


Figure 5.5: Comparison of time and function evaluations required with varying number of sampling points

Chapter 6

Conclusion and Discussion

1. Number of Points: For building a surrogate model before optimization step, the number of data points to be used is a hyper parameter. A comparison between using 10, 20, 40, 60 and 80 points is shown in figure 5. The number of function evaluation and the time taken to reach the optimal point for a black box function camel6 is least for 20 number of points.

2. Sampling Method : Covering the input space properly is an important part of the problem so as to build a surrogate model which is quite a good approximation of the actual model. Four different sampling methods were considered and the progress of the result with them is observed. Sobol-sequence sampling method took least time and used the least number of function evaluation as compared to other methods.

3. Algorithm : The basic algorithm includes building a surrogate model and optimizing it while continuing this procedure until convergence. After finalizing the sampling method and the number of points to be used for sampling, the next thing is to chose between global and local search.
In global optimization method, inputs are sampled all over the space, Thus the surrogate model built using this is not a very good approximation in most of the cases. For 2 variable problems, global optimization fails to converge to the solution even for around

3000 function evaluations.

An improvement of this could be to find a local search method. Trust region method is one such method which builds a trust region around the point and optimize in that region. This algorithm converges to result for some of the cases but fails to converge as the dimensionality of the problem is increasing.

To deal with multiple dimension, an algorithm called cyclic coordinate search method is used. Combining it with trust region method gives us a new algorithm, which takes one dimension at a time and a local region. Sampling and optimization in that region in one dimension and then proceeding to the next dimension. This algorithm is very effective and succeeded to give results for around 80 percent of the problem of up to 10 dimension. For more than 10 dimensions, even this algorithm failed to converge or it was taking too long to converge.

Bibliography