

# Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

## REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS


FOR THE DEGREE OF Master of Science

TITLE Integration of Coordinate Descent and Machine Learning for Black-Box  
Optimization

PRESENTED BY Swapnil Agrawal

ACCEPTED BY THE DEPARTMENT OF

Chemical Engineering

  
NIKOLAOS SAHINIDIS, PROFESSOR AND ADVISOR

12/4/2019  
DATE

  
ANNE S. ROBINSON, PROFESSOR AND DEPARTMENT HEAD

12/4/2019  
DATE



# Integration of Coordinate Descent and Machine Learning for Black-Box Optimization

MASTER OF SCIENCE DEGREE RESEARCH PROJECT

December 2019

Swapnil Agrawal

Advisor - Nicholas V. Sahinidis

Department of Chemical Engineering

# Contents

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
<b>3</b>	<b>Literature Survey</b>	<b>6</b>
<b>4</b>	<b>Proposed Algorithm</b>	<b>9</b>
4.1	Sampling methods . . . . .	9
4.2	Building Surrogate Model . . . . .	13
4.3	Search Algorithms . . . . .	14
<b>5</b>	<b>Results and Discussions</b>	<b>17</b>
5.1	Experimental Setup . . . . .	17
5.2	Test Problems . . . . .	17
5.3	Hyper Parameter Tuning . . . . .	19
5.4	Solving all black-box functions . . . . .	24
<b>6</b>	<b>Conclusion</b>	<b>27</b>
	<b>Bibliography</b>	<b>29</b>

# List of Figures

3.1	Trust Region Method in 2-dimension . . . . .	7
3.2	Coordinate Descent Method in 2-dimension . . . . .	8
4.1	Random Sampling in 2 Dimensions . . . . .	10
4.2	Sobol Sampling in 2 Dimension . . . . .	11
4.3	Latin HyperCube Sampling in 2 Dimension . . . . .	12
4.4	Hammersley Sampling in 2 Dimension . . . . .	13
4.5	Cordinate Descent Method Flow Chart . . . . .	16
5.1	Characteristic of test problems . . . . .	18
5.2	Distributions of problems by dimensionality and classes . . . . .	18
5.3	Distributions of problems by dimensions . . . . .	19
5.4	Solution Progress with Different sampling methods . . . . .	20
5.5	Solution Progress for Varying number of sampling points . . . . .	21
5.6	Time and Function evaluations with varying number of sampling points .	22
5.7	Progress of solution for global search . . . . .	23
5.8	Progress of solution for local search . . . . .	23
5.9	Fraction of problems solved with global and local search for 4 categories .	24
5.10	Fraction of problems solved with local search . . . . .	25
5.11	Fraction of problems solved with global search . . . . .	26

# Chapter 1

## Abstract

As the system are becoming more complex, it is becoming easier to experiment rather than understanding them. In other cases, the function is not at all available in algebraic form though we have the access to output on providing it some input. Hence, solving black-box function optimization problem is gaining importance. The project aims to perform optimization by integrating coordinate descent and machine learning methods. Three different search algorithms 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 245 non-convex smooth black box problems. Practical scalability 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 finding the minimum energy required for a process or calculating hyper-parameter for a machine learning algorithm such as depth in a neural network.

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 245 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. [13]

Previous researchers have approached black-box optimization using meta-heuristic methods such as Particle Swarm Optimization (PSO) [7], Ant Colony Optimization etc. These methods usually require more evaluations of black-box function and are expensive [5].

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 ([4], [6], [8]) for black-box problems of low effective dimensionality. But for higher dimensional problems, Bayesian Optimization becomes too complex to solve. In this project, we aim to solve non-convex black-box function and to explore practical scalability of the methods with respect to the dimension of the search space and non-linearity.

We have tested 3 different algorithm [9] on a black box function camel6. 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  $[0.0898, 0.7126]$  [13].

These algorithms use Coordinate Descent [14] and Trust Region methods[12] and are described in the next chapters in detail.

### Trust Region Method

In Trust-region method, a region is defined 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 optimum value that model depicts within the region. The size of the trust region in each iteration would depend on the improvement previously made.

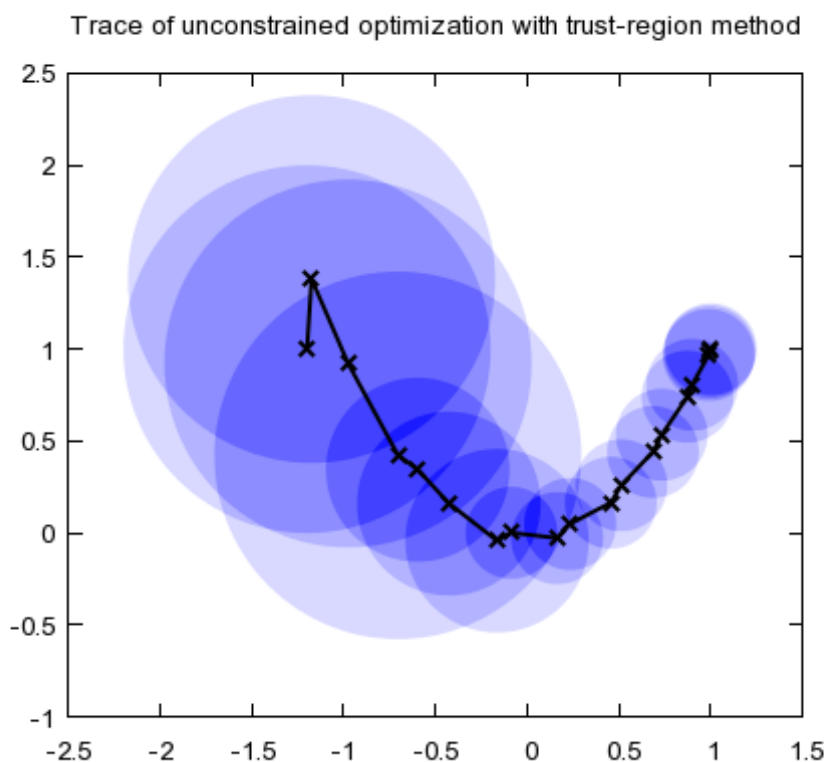


Figure 3.1: Trust Region Method in 2-dimension

### Coordinate Descent Method

It is an optimization algorithm that successively minimizes along the coordinate directions to find the minimum of a function. At each iteration, the algorithm determines a

coordinate or a coordinate block [11], then exactly or inexactly minimizes over the corresponding coordinate hyper plane while fixing all other coordinates or coordinate blocks [14].

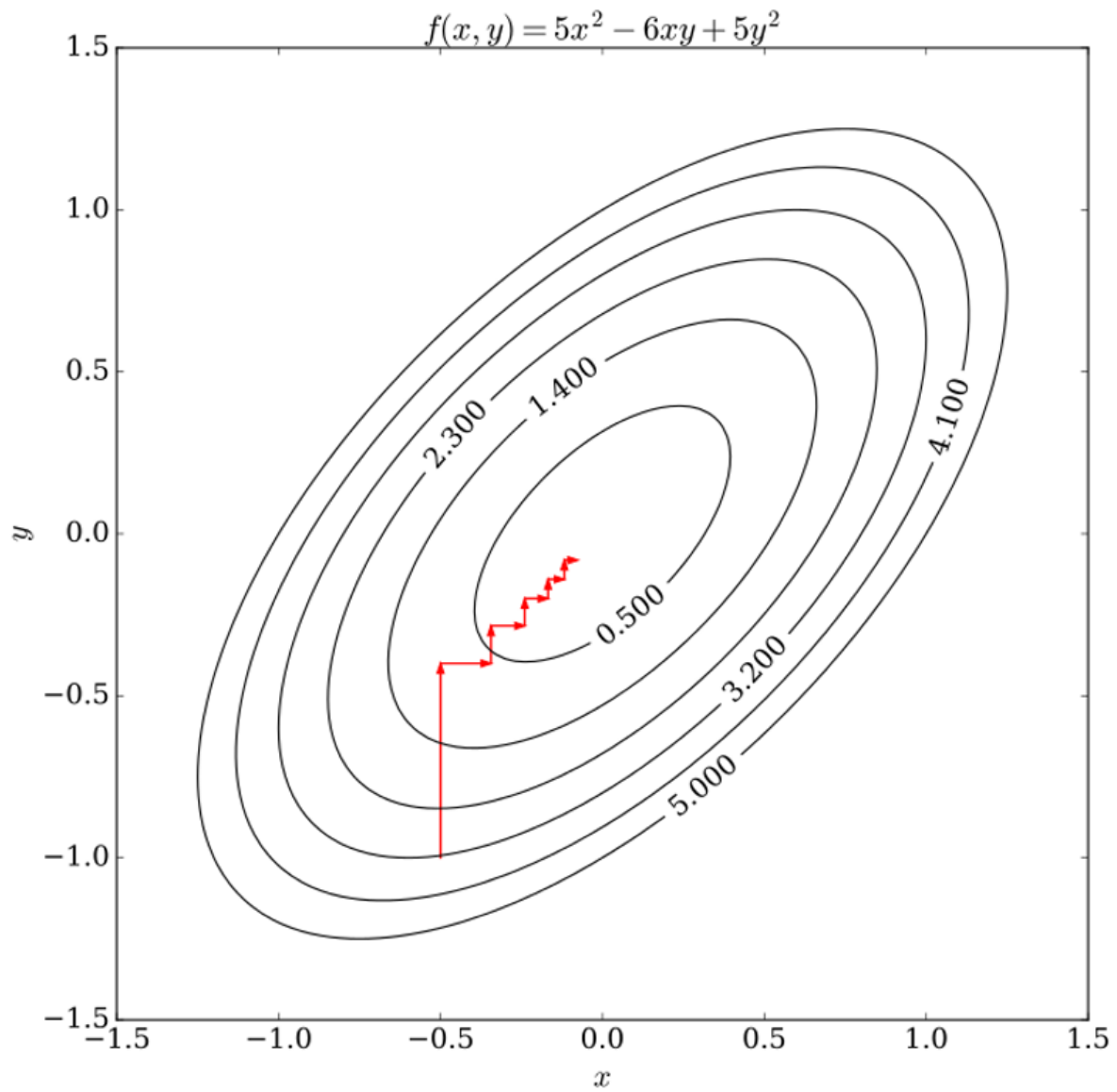


Figure 3.2: Coordinate Descent Method in 2-dimension

# Chapter 4

## Proposed Algorithm

For solving black-box optimization problem, the overall approach in this project is to select a region, sample points in the selected region using a sampling method, fit these points into an algebraic function using machine learning or ALAMO and then optimize using BARON. The process is repeated until we have reached a stopping criteria or have achieved convergence.

Using a different combinations of sampling methods, sampling region and the update rules, different final results can be obtained. 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. Four 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. Every 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 areas having important information about the function.

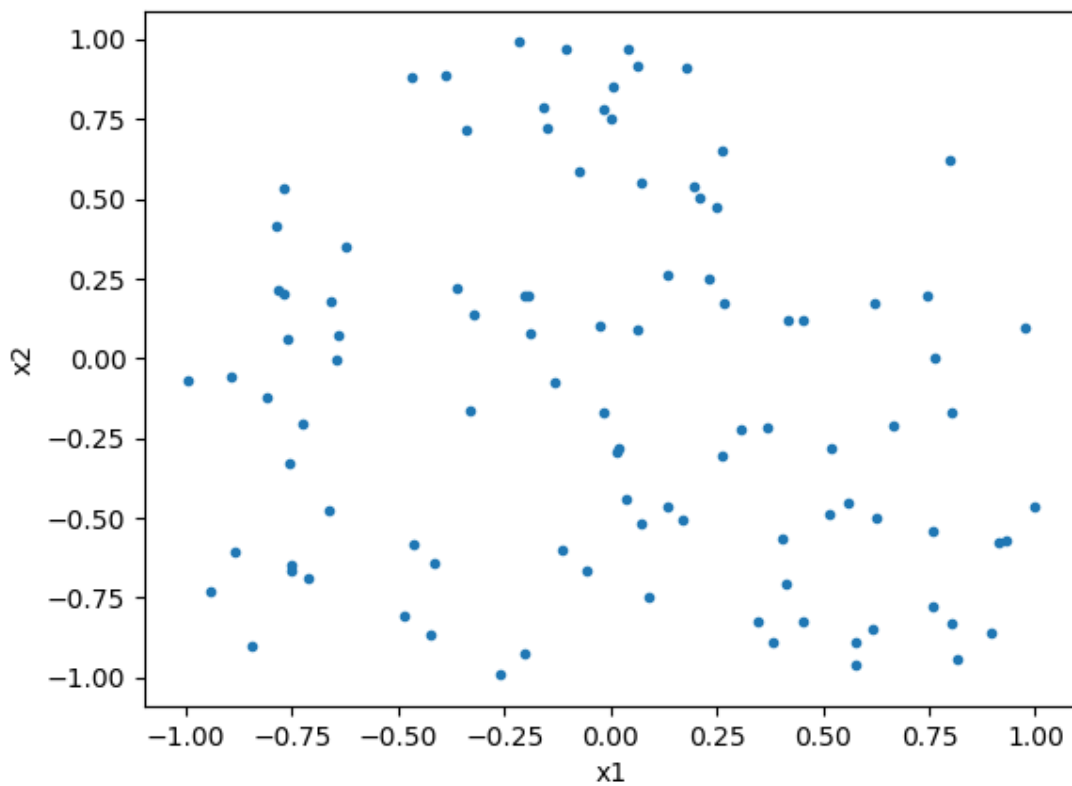


Figure 4.1: Random Sampling in 2 Dimensions

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 [3] shows sobol sequence for a sampling size of 100 points in 1 dimension.

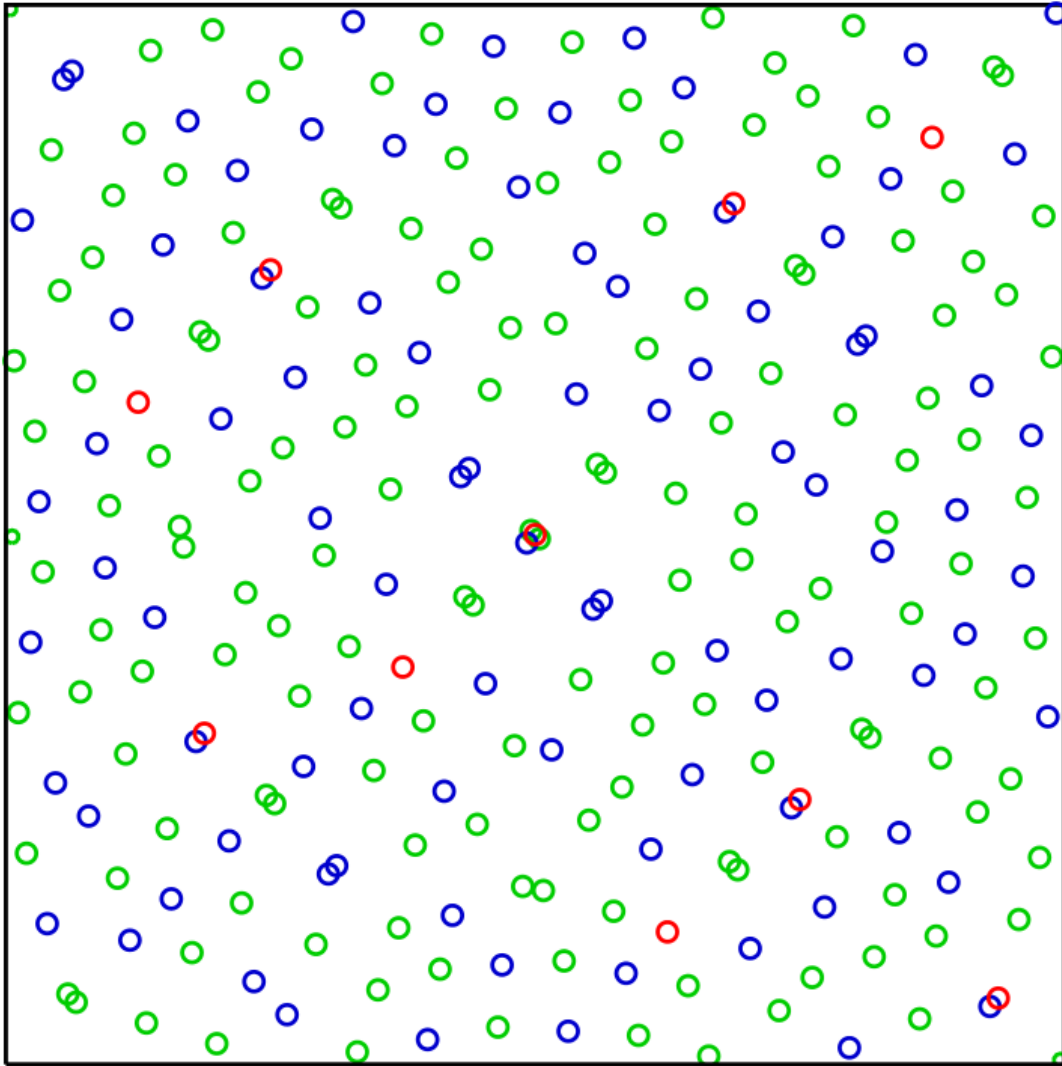


Figure 4.2: Sobol Sampling in 2 Dimension

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. [2]

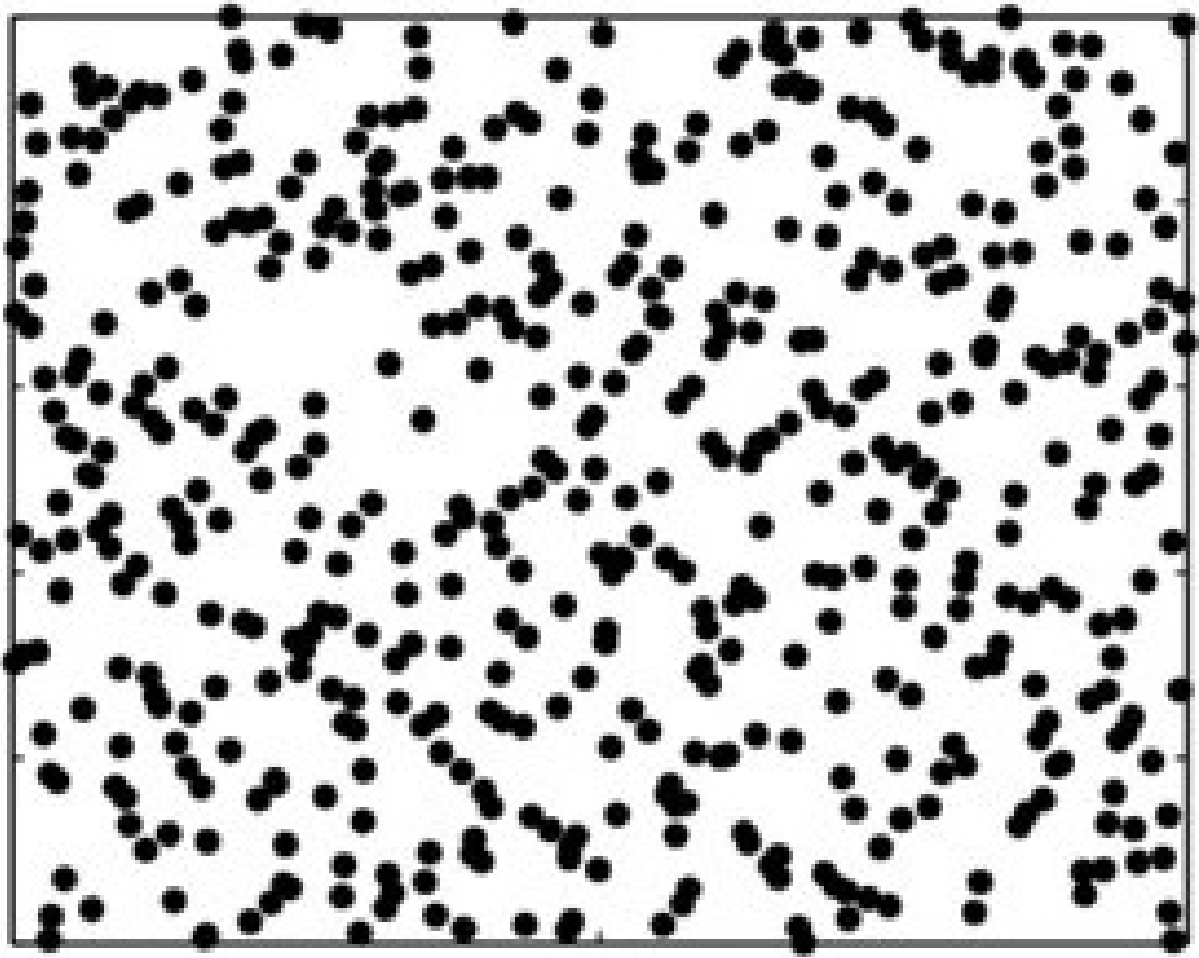


Figure 4.3: Latin HyperCube Sampling in 2 Dimension

4. Hammersley : Hammersley is also a quasirandom sequence same as Sobol except that here the van der Corput sequence is generalized to higher dimension to generate sample. [1]

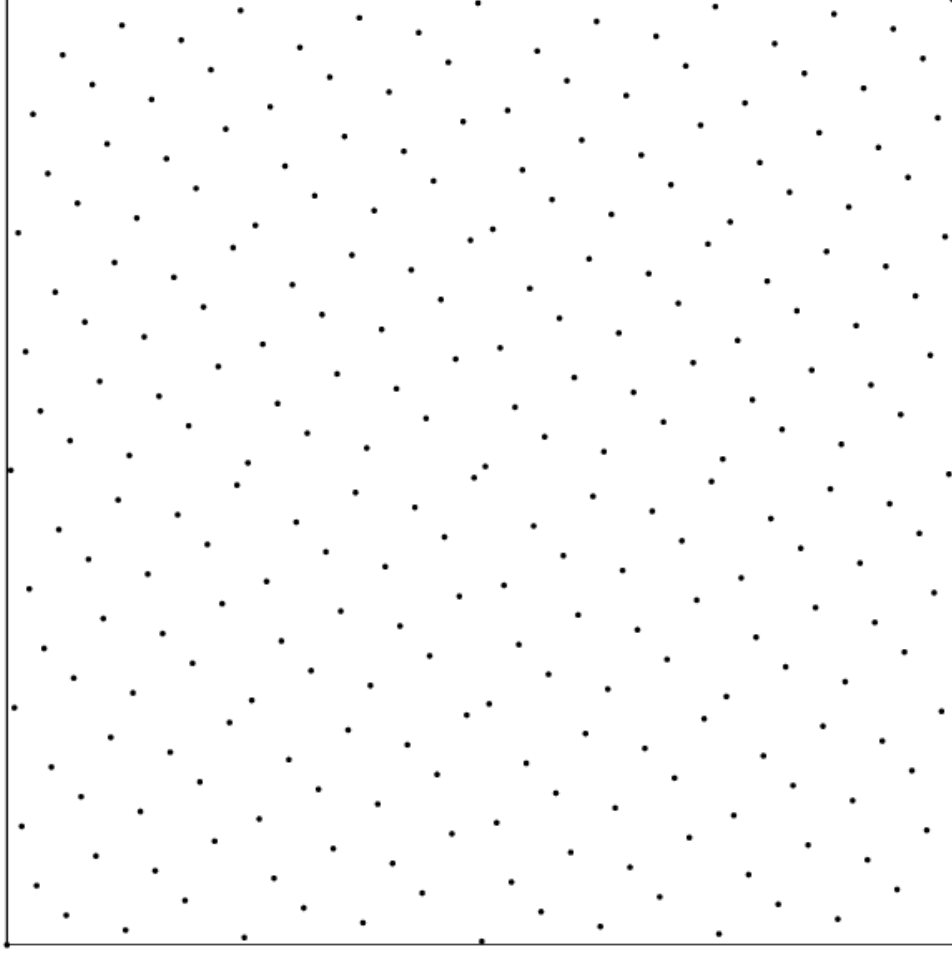


Figure 4.4: Hammersley Sampling in 2 Dimension

## 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 data set, the machine learning algorithm from the package can be used to build a surrogate model. Along with the model param-

eters, it also provides details about the model performance over a validation data set.

## 4.3 Search Algorithms

### 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 the output for all 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 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 = 1e7 (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 the 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 points along that coordinate
  - Compile black-box function and Calculate output
  - Build a surrogate model
  - Use this surrogate model to find function optima along that direction
  - Update stored optimum value (z0) only if there is an improvement
  - if (improvement in optimum value) : shrink region
  - else: expand region
  - niter = niter + 1
-

We will use two variants of cyclic coordinate search for testing on black-box function. One with global search where the search region is not changed and one where the region is changed as described in above algorithm. The same is depicted in the following process flow chart.

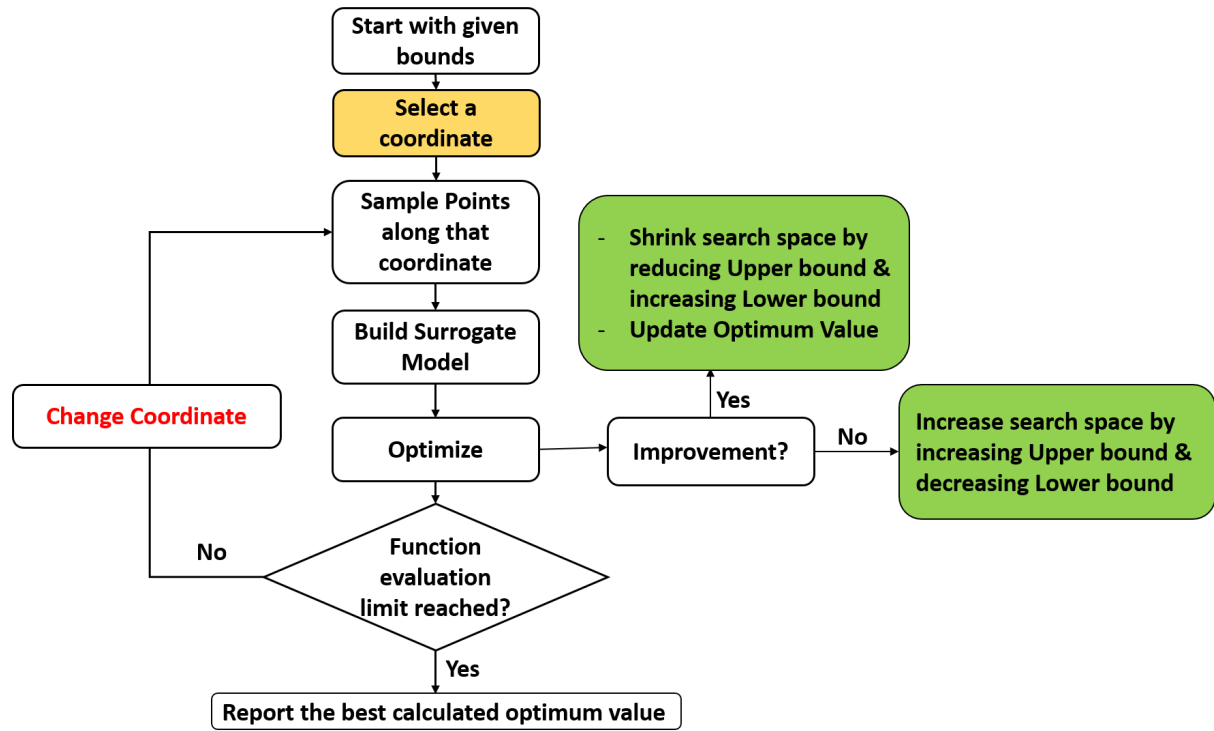


Figure 4.5: Cordinate Descent Method Flow Chart

# Chapter 5

## Results and Discussions

### 5.1 Experimental Setup

All computations are performed on Intel 2.7 GHz processors running Windows and Python 3.7.2.

245 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 from [13]. The various hyper-parameter such as number of sampling points in one iteration, sampling method, search space 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 the obtained result is 13.74 KW/tonne/day whereas the known optimum value which is 10.99 KW/tonne/day [10]

### 5.2 Test Problems

A set of 500 black-box problems was available 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 ( $n_{avg}$ ) equal to 37.6 [13]. Table and Figure below from [13] presents the distribution of problems by dimensionality and by problem classes.

$n$	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

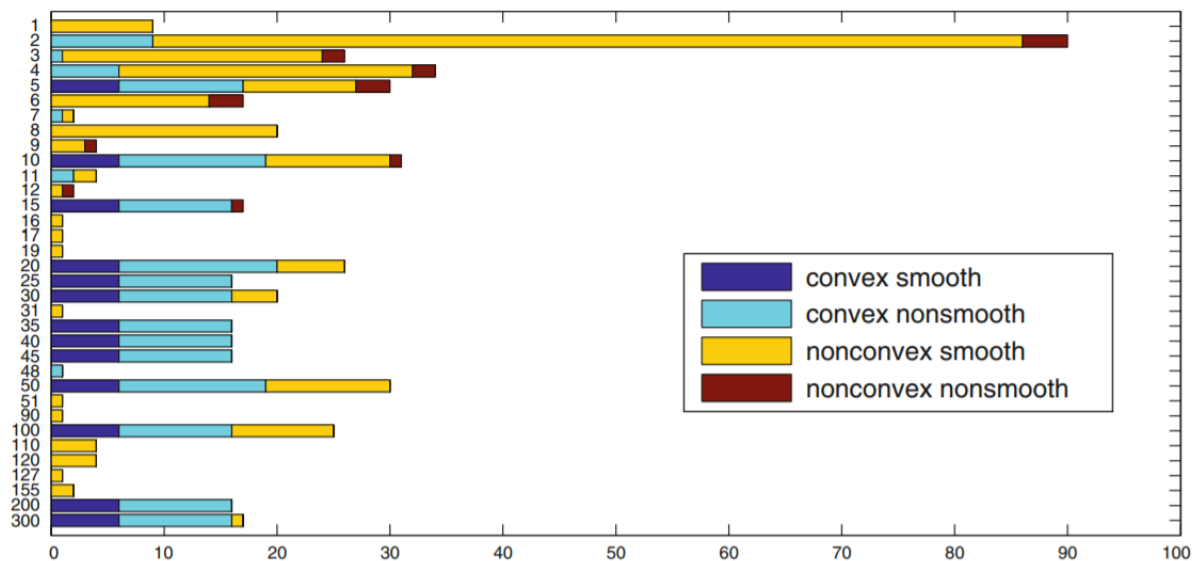


Figure 5.2: Distributions of problems by dimensionality and classes

Out of these 500 problems, non-convex smooth problem for selected for the analysis of the algorithm in this project. These problems are further divided into 4 categories based on the number of variables to compare the scalability of the algorithm as shown in the table below.

Category	Number of Variables	Total Problems
1	1-2	86
2	3-9	97
3	10-30	27
4	>30	35

Figure 5.3: Distributions of problems by dimensions

### 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 black-box function is tested and the performance is compared by varying these parameters. The detailed discussion of this problem is mentioned in [13], hence it is easy to compare results for hyper-parameter tuning.

**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.

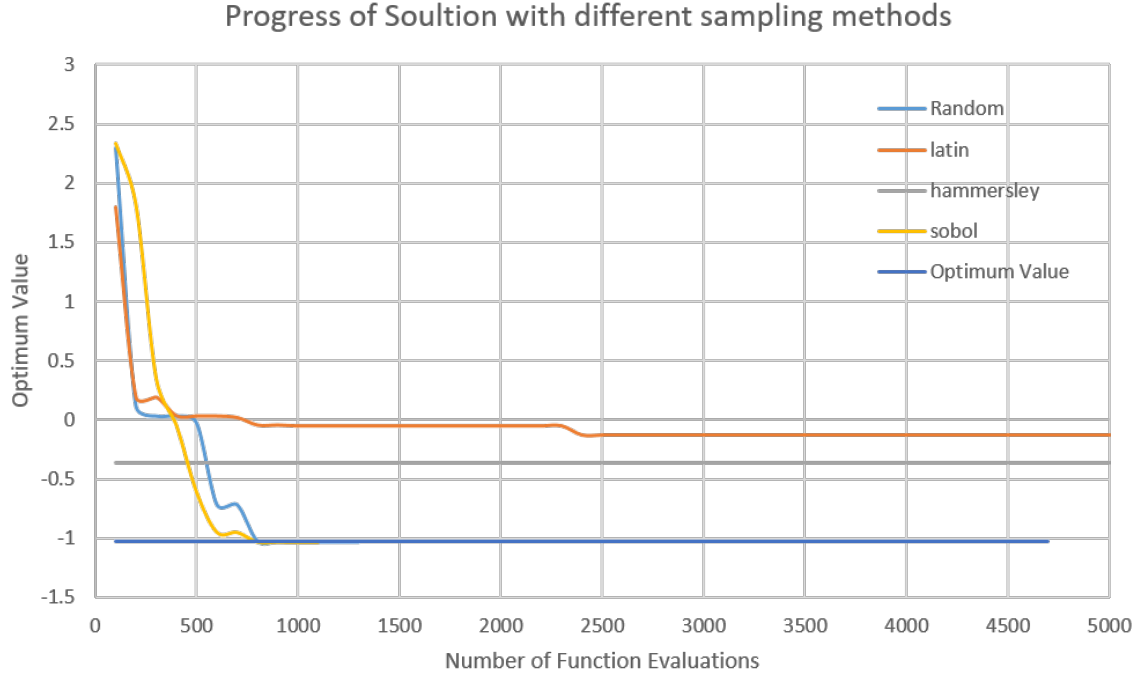


Figure 5.4: Solution Progress with Different sampling methods

Sobol-sequence method reached the optimum value in the least number of iterations among all sampling methods. Thus we will use sobol sequence for further calculations.

**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.

Solutions with taking 20 points in one iterations are able to reach to the optimum value fastest and required least number of function evaluation to converge.

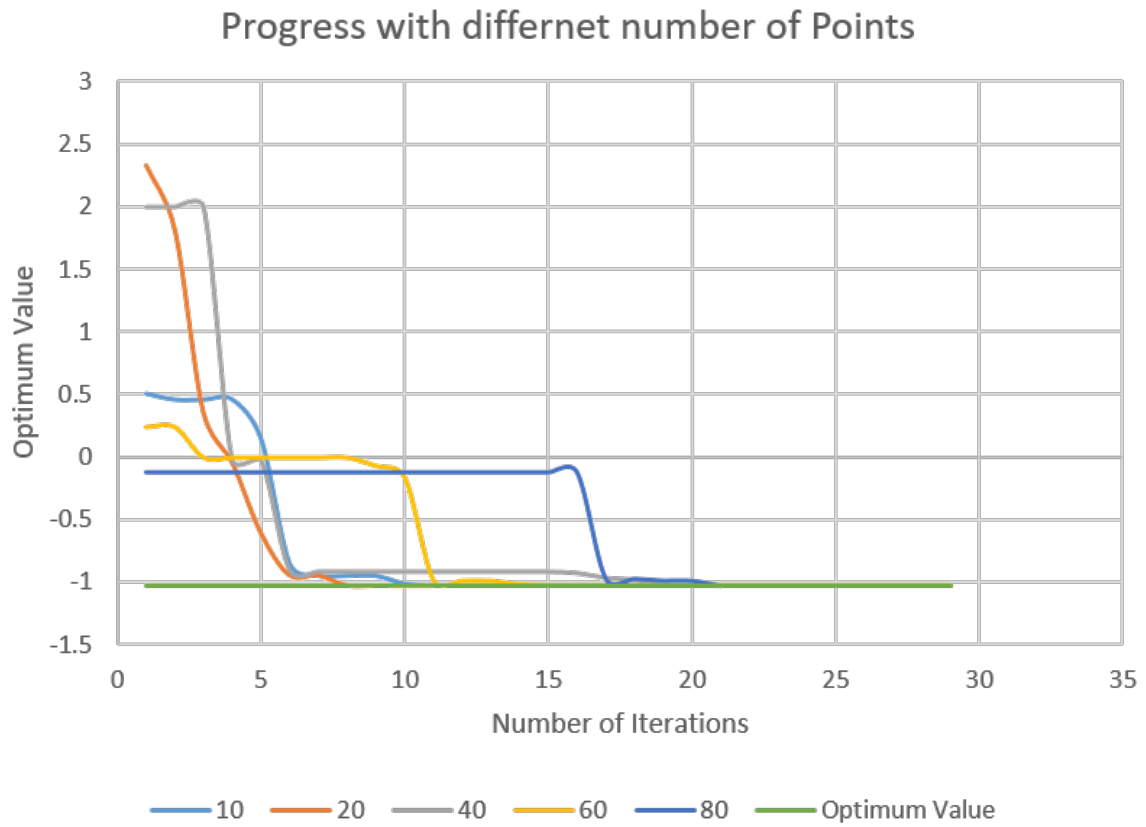


Figure 5.5: Solution Progress for Varying number of sampling points

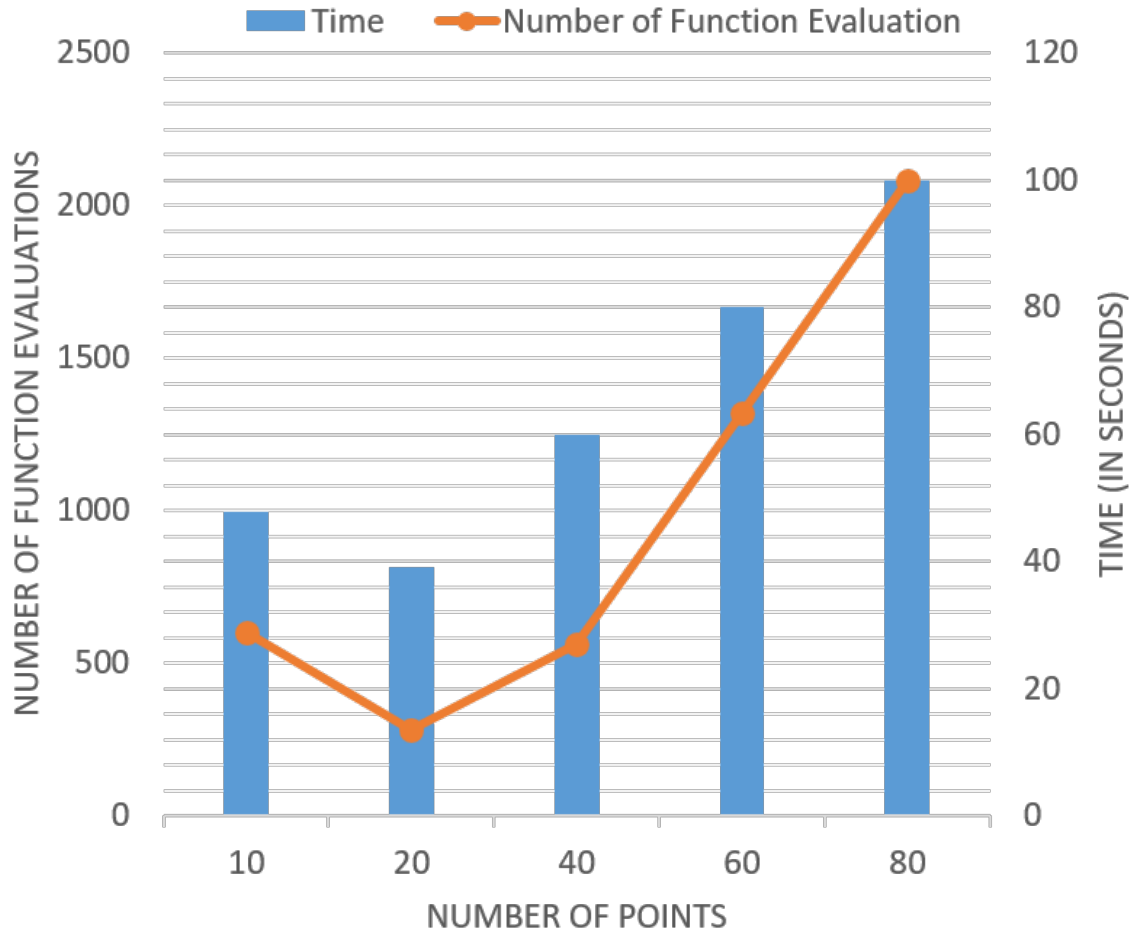


Figure 5.6: Time and Function evaluations with varying number of sampling points

**3. Global Vs Local** Camel 6 black-box function is solved with coordinate descent method with global and local search space. The progress of result with iteration for both search space is shown below. The optimum value progresses faster with global search in the beginning and reached near optimum value faster but then stopped improving after that. The local search on the other hand is slower in the beginning and takes time to reach optimum value but it did converged within 2500 function evaluations.

The proposed solution for this is to use global search in the beginning and then start performing local search after few iterations.



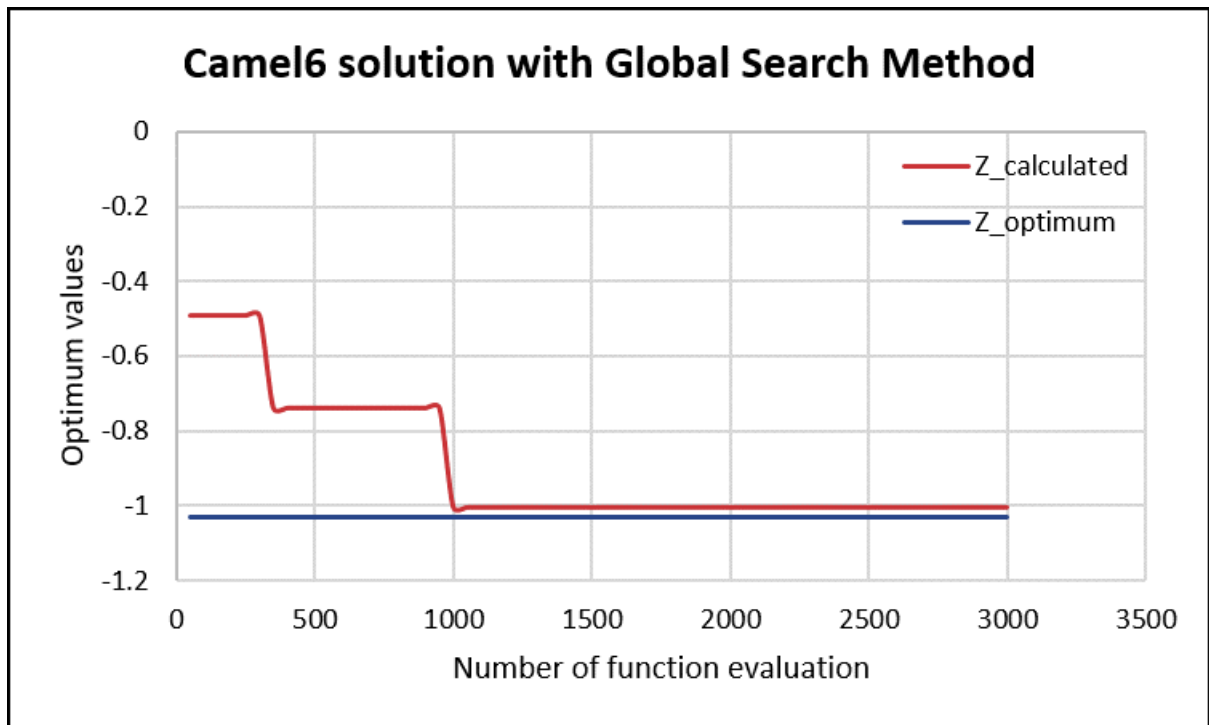


Figure 5.7: Progress of solution for global search

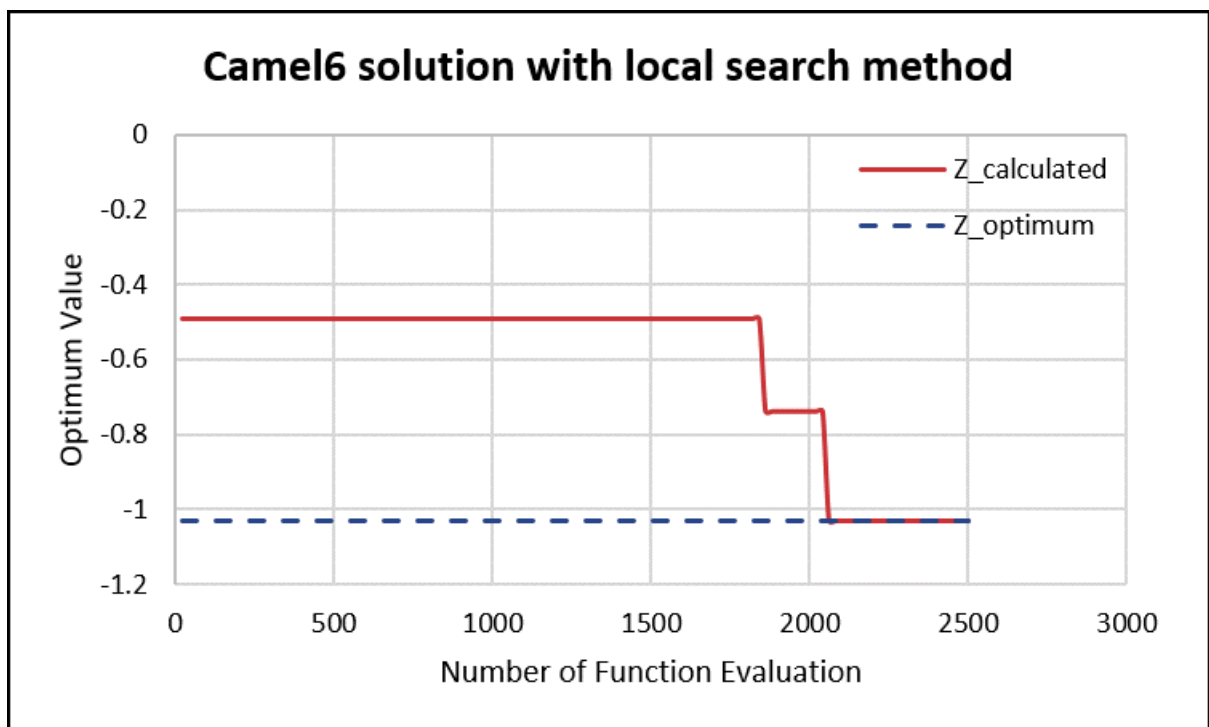


Figure 5.8: Progress of solution for local search

## 5.4 Solving all black-box functions

All non-convex smooth black-box function are solved with coordinate descent method and the optimized hyperparameter. Both global and local search space is used to further compare their performance. The result of fraction of problems solved for each category is shown below.

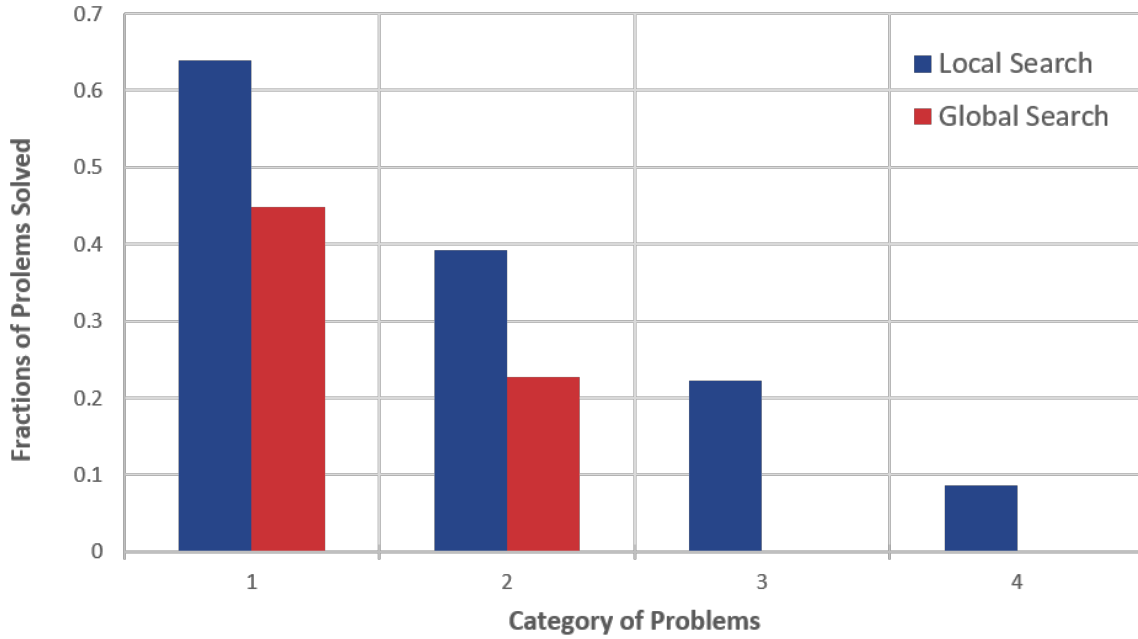


Figure 5.9: Fraction of problems solved with global and local search for 4 categories

The fraction of problems solved for category 1, i.e. lower dimensional problems is higher for both global and local search. As the dimensions increases, the fractions decreases for both the methods. This is due to the simple nature of our algorithm which is using one dimension at a time and is not able to solve high dimensional problems.

Also, compared to local search, the fraction of problems solved with global search is less. This is because approximating a function in a larger area is difficult as compared to approximating it in smaller region.

The results for fraction of problems solved with the number of function evaluations are shown in figure below. It can be observed that after 2500 function evaluations, the fraction of problems solved saturates for all the categories.

Overall, local search was able to solve more problems compared to global search and as the dimension of the problems increases, the performance of both global and local search decreased by a major fraction.

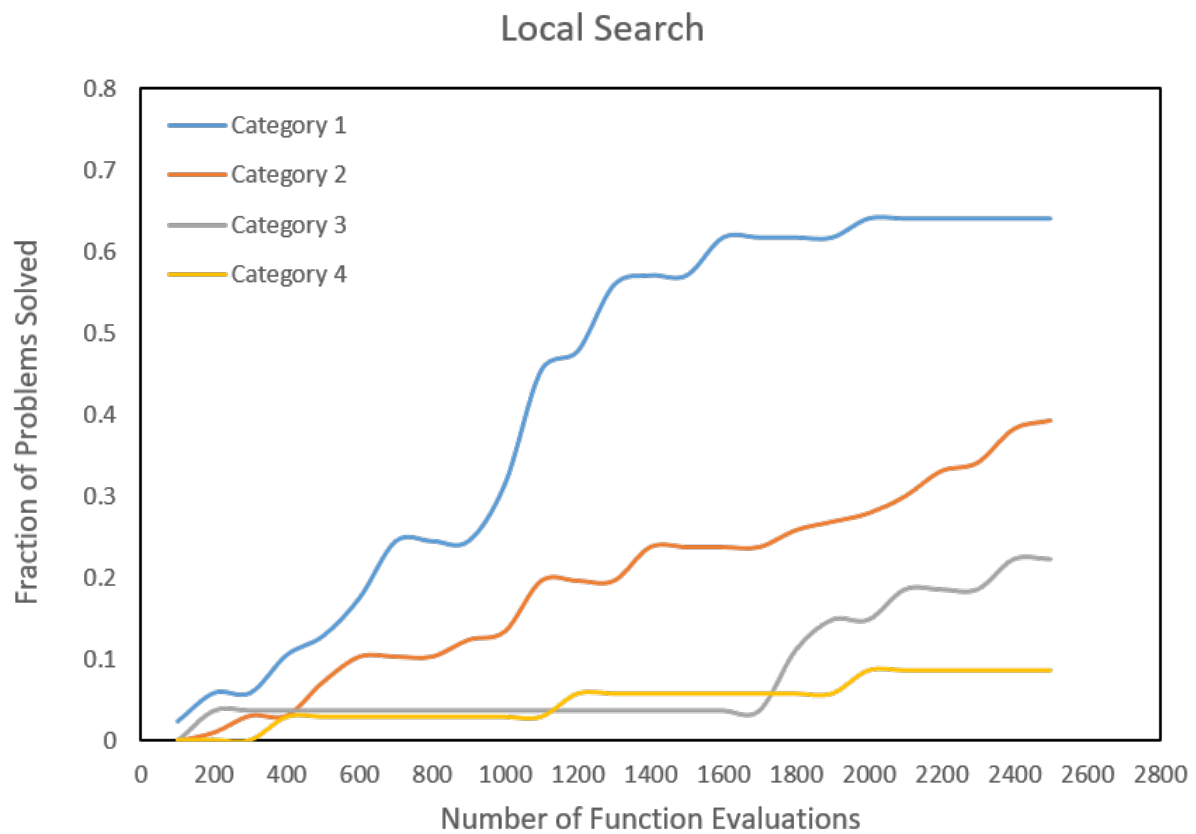


Figure 5.10: Fraction of problems solved with local search

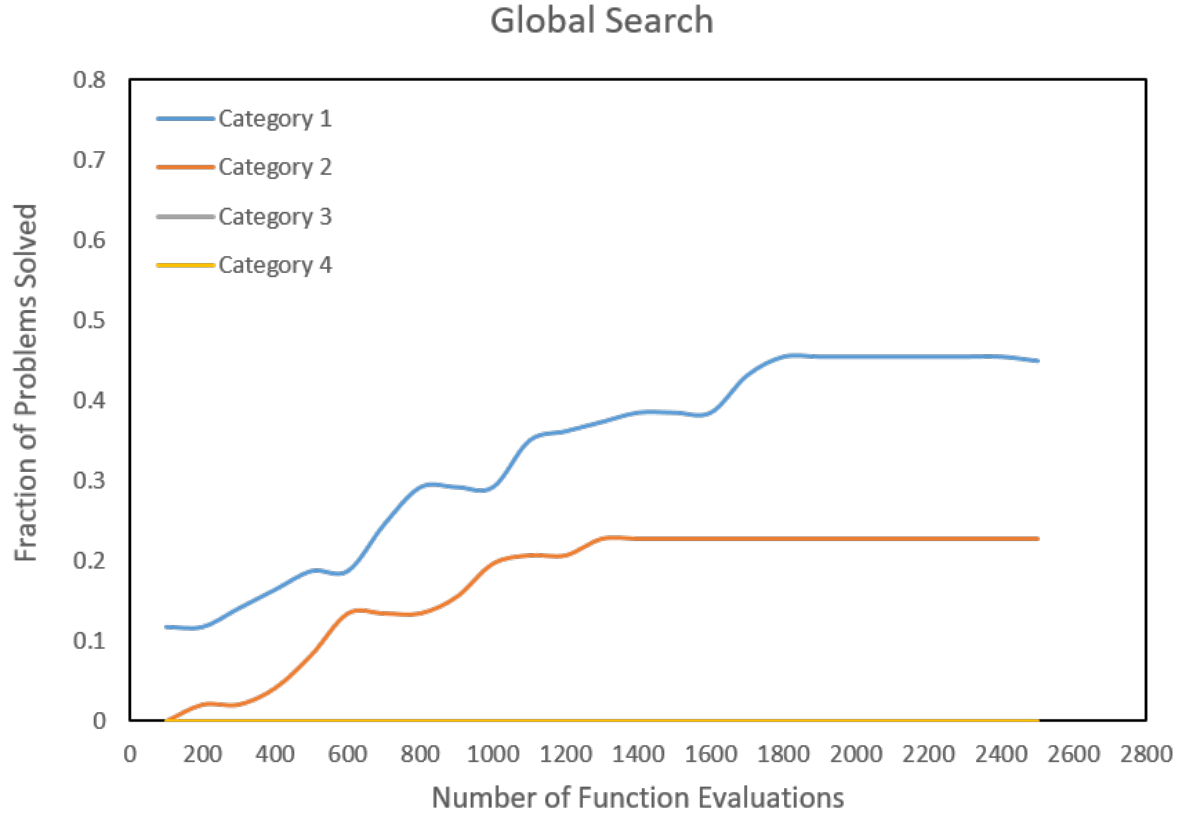


Figure 5.11: Fraction of problems solved with global search

The algorithm is tested on a pilot Aspen Black-Box function. The objective is to optimize the energy of Single mixed refrigerant (SMR) process which is modeled by Aspen HYSYS. The problem has 8 variables in total with linear and non-linear constraints. With 20,000 function evaluation and 2 cpu days, the optimum value of 13.74523 KW/(ton/day) is obtained whereas the known optimum value for the problem is 10.99 KW/(ton/day) by [10]. The algorithm is not able to perform well for this problem as it was designed to address only bound constraints whereas this aspen problem also contains few hidden constraints.

# Chapter 6

## Conclusion

Non-convex smooth black-box functions with dimensions ranging from 1 to 300 are optimized using Coordinate Descent Method. Global and Local Search space are used and their performance over these black-box functions are compared. From the results of fractions of problems solved for both global and local search, it can be concluded that coordinate descent should be a very simple model for problems more than 30 dimension and a more complex model should be used instead.

For lower dimensional problems, if the function evaluation limit is less, global search should be used as it can reach near optimum in less function evaluations whereas if function evaluation limit is more, using local search will provide better optimum value.

Lastly, the algorithm was only accounting for variable bounds constraints. For solving the aspen problem, the algorithms needs to be changed to address the hidden constraints as well.

# Bibliography

- [1] Hammersley - the hammersley quasi monte carlo (qmc) sequence. [https://people.sc.fsu.edu/~jburkardt/py\\_src/hammersley/hammersley.html](https://people.sc.fsu.edu/~jburkardt/py_src/hammersley/hammersley.html).
- [2] Latin hypercube sampling. <https://mathieu.fenniak.net/latin-hypercube-sampling/>.
- [3] Graud Blatman, Bruno Sudret, and Marc Berveiller. Quasi random numbers in stochastic finite element analysis. *http://dx.doi.org/10.1051/meca:2007051*, 8, 05 2007.
- [4] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.
- [5] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018.
- [6] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.
- [7] James Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2010.
- [8] Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994.

- [9] Mario A Muoz, Yuan Sun, Michael Kirley, and Saman K Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.
- [10] Jonggeol Na, Youngsub Lim, and Chonghun Han. A modified direct algorithm for hidden constraints in an lng process optimization. *Energy*, 126:488–500, 2017.
- [11] Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.
- [12] Trust Region. Trust-region methods - optimization. [https://optimization.mccormick.northwestern.edu/index.php/Trust-region\\_methods](https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods).
- [13] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, Jul 2013.
- [14] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.