Pruned Search: A Machine Learning Based Meta-Heuristic Approach for Constrained Continuous Optimization

Ruoqian Liu, Ankit Agrawal, Wei-keng Liao, Alok Choudhary
EECS Department
Northwestern University
Evanston, IL USA
{rll943,ankitag,wkliao,choudhar}@eecs.northwestern.edu

Zhengzhang Chen
NEC Laboratories America, Inc.
Princeton, NJ USA
zchen@nec-labs.com

***Abstract*—Searching for solutions that optimize a continuous function can be difficult due to the infinite search space, and can be further complicated by the high dimensionality in the number of variables and complexity in the structure of constraints. Both deterministic and stochastic methods have been presented in the literature with a purpose of exploiting the search space and avoiding local optima as much as possible. In this research, we develop a machine learning framework aiming to 'prune' the search effort of both types of optimization techniques by developing meta-heuristics, attempting to knowledgeably reorder the search space and reduce the search region. Numerical examples demonstrate that this approach can effectively find the global optimal solutions and significantly reduce the computational time for seven benchmark problems with variable dimensions of 100, 500 and 1000, compared to Genetic Algorithms.**

***Keywords*-constrained optimization; complexity reduction; machine learning; meta-heuristics**

I. Introduction

Searching for suitable solutions that optimize an objective function, and/or satisfy a set of constraints, is a general technique and ubiquitous solution to applications in various engineering areas. Search problems are established around three components: variables, constraints, and the objective function, and searches are performed in the domain of each variable which can be either continuous or discrete. Discrete optimization, or combinatorial optimization, has a finite set of solutions and is usually represented by graph structures and approached with heuristic search (e.g. A*) and dynamic programming. Continuous optimization, however, involves search spaces that are infinite, and the challenge of search escalates when high dimensionality in the number of variables and great complexity in the structure of constraints are acknowledged. A d-dimensional continuous optimization problem can be expressed as:

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq b_i, i = 1, \ldots, n. \end{aligned}$$

Here $x = (x_1, \ldots, x_d)$ is the vector of variables, the function $f_0 : \mathbb{R}^d \to \mathbb{R}$ is the objective function, and functions $f_i : \mathbb{R}^d \to \mathbb{R}, i = 1, \ldots, n$, with constants $b_1, \ldots, b_n$, are the constraints. In computational geometry, such a problem has a problem domain (if feasible) as a polyhedra in $\mathbb{R}^d$, defined by the n halfspaces from the n constraints.

Excluding brute-force methods where making exhaustive enumeration of solutions is insurmountable for continuous problems (unless some discretization is performed), classical algorithmic optimization techniques can be divided into two main groups. The first group of methods are searches typically based on deterministic numerical rules realized through the differential calculus, such as Gradient Search, Linear Programming, Quadratic Programming, etc. They are deterministic in that given the same initial values, repeated runs will always produce the same result. In practice, this quality renders overcoming local optima a serious problem. As an alternative, the other group of methods introduces stochastic elements. Examples including Simulated Annealing, Genetic Algorithms [1] and Particle Swarm Optimization are less likely to get stuck in local optima are and more flexible towards discovering new solution spaces.

Our work in [2] has proposed the concept of search space preprocessing to narrow down the search space which could then be used by traditional searches, in order to ease the curse of dimensionality. The framework consisting of dataset construction, search order reduction and search domain reduction was proposed, employing data mining and dimension reduction techniques to gain informative insights about the problem space, so as to direct the search quest into more promising areas. In this paper, we implement the proposed idea into 'Pruned Search', a Machine Learning (ML) based meta-heuristic developed for fast and accurate optimization search for constrained and continuous problems.

Given an optimization problem with a large amount of variables as well as constraints, we first collect a set of representative variable-objective data instances using the proposed data distillation technique based on vertex enumeration and Lagrangian relaxation. Then, by employing feature selection and classification techniques, we refine the search path and search region. Finally, a simple line search-like algorithm is proposed to enhance the optimization. The rest of the paper is organized as follows. Section II goes into the methodology of the proposed framework, with a subsection dedicated to each of our three key processes. In Section III we present experiments conducted on: (1) a simple problem to validate path refinement, (2) a number of canonical test problems, and

(3) a synthetic problem. Section IV lists related works, and Section V concludes the paper.

II. Methodology

A. Overview

Following the philosophy stated in [2] we consider developing meta-heuristics with ML to enhance the search process in optimization. The attempt is to have the search force focused in a more promising path and prune the irrelevant effort. In an optimization problem, variables cannot be fumbled freely with dimension reduction methods such as PCA, but we can still assume that one of the following statements will hold.

- Assumption 1: The desired (optimal) value of the function depends only on a reduced, albeit unknown, set of variables.
- Assumption 2: The impact of each variable to the function is different. Hence, there exists an optimal order in terms of searching priority.

The first assumption imposes an additional parameter—the size of the subset of "active" variables. Both assumptions are commonly seen in many industrial processes, during the design of which, variables are included as many as possible but not all of them are equally important. The two assumptions together complete the scene of intrinsic variable priority on which our proposed method is based.

In this setting we can apply feature selection in data mining to analyze variable relations. The framework contains three major components, as shown in Fig. 1. The functionalities of each component are introduced below, and further explained and illustrated with an example in the following sections.

Data distillation. For ML to work, it needs to have access to a set of data that contain values of the variables and the corresponding function value. The data has to be distinguishably significant in that it contains instances with the most wanted (highest or lowest corresponding to maximization or minimization) function values. We obtain such a distilled significant data set by firstly transforming the problem into the computational geometry regime to form a polyhedra representing the feasible space, and then extracting the vertices of the polyhedra. Lagrangian relaxation is used to handle complicated constraining conditions.

Complexity reduction. With a proper collection of variable-objective value instances, ML can learn to extract information of variables to produce two functionalities: 1) the creation of an ordered list of variables based on their influence and impact towards the function, and 2) the reduction of the feasible region in variable searching. The former is achieved through feature selection methods where a ranking is guaranteed. The latter is realized through examining a rule-based classifier and looking for the critical thresholds.

Enhanced optimization. Optimization becomes a much promising endeavor when the search is "pruned", in that the space is reduced and a pre-planned searching path is deployed. A simple line search-like algorithm is employed. We specify a prefixed searching order, and replace the original constraints with the pruned ones.

We will use an example problem throughout Section II-B to Section II-D to illustrate each step. The example problem, G1, is taken from [3]. It is formulated as follows.

G1 Problem:

$$
\begin{aligned}
\text{Min:} \quad & f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
\text{s.t.:} \quad & g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0, \\
& g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0, \\
& g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0, \\
& g_4(x) = -8x_1 + x_{10} \leq 0, \\
& g_5(x) = -8x_2 + x_{11} \leq 0, \\
& g_6(x) = -8x_3 + x_{12} \leq 0, \\
& g_7(x) = -2x_4 - x_5 + x_{10} \leq 0, \\
& g_8(x) = -2x_6 - x_7 + x_{11} \leq 0, \\
& g_9(x) = -2x_8 - x_9 + x_{12} \leq 0, \\
& x_i \geq 0, i = 1, \ldots, 13, \\
& x_i \leq 1, i = 1, \ldots, 9, 13, \\
& x_i \leq 100, i = 10, 11, 12.
\end{aligned}
$$

The G1 problem has 13 variables, 9 complex linear combinational constraints and 2 boundary constraints on each of the variable, in total 35. The true global minima is known as $x^* = (1, 1, \ldots, 1, 3, 3, 3, 1), f(x^*) = -15$. G1 is not chosen to prove the robustness of our methodology to high dimensions (which will be exhibited in Section III with problems up to 1,000 dimensions), but rather to help illustrate each step.

B. Data Distillation with Vertex Enumeration

Given an optimization problem, the simplest way of data collection would be feeding in random combinations of valid variable values to the function and collecting its outputs. Number-theoretic methods (NTM's) are a class of techniques by which representative points of the uniform distribution on the unit cube can be generated [4]. One example of NTM is the quasi-Monte Carlo model.

However, while a uniformly randomized set could be representative for numerical analysis of functions, it does not necessarily serve as representative when it comes to machine learning purposes. Just as we only need to show a child red and green cards if what we want him/her to learn is to distinguish red and green, we claim a representative and significant set for analyzing variable relations is the collection of instances that produce very high and very low function values. One set is the "target", and the other is included as an opposing force. Depending on what feature selection method is used, the opposing set can sometimes be omitted.

To collect such a data set, we follow the idea of vertex enumeration in computational geometry. The set of linear inequalities $Ax \leq b$ defines a polyhedron in R^d. Each of the n inequalities represents a bounding hyperplane that together constitutes the polyhedra. An intersect of any d inequalities forms a vertex. The objective function $f(x)$ also represents a hyperplane in R^d, and the optimum answer is one of the vertices on this plane. In fact, all the vertices of the polyhedra are extreme values to the function. Therefore, to obtain a set

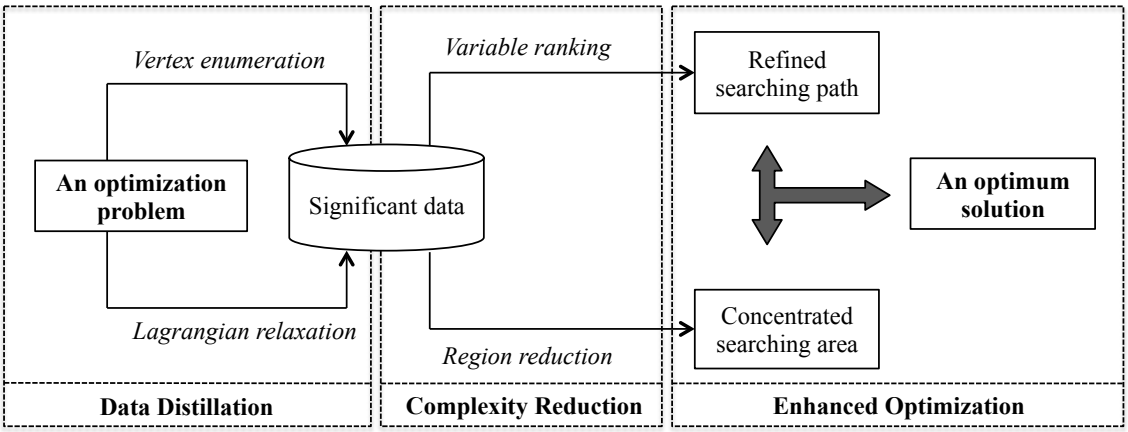

Fig. 1: Overview of the proposed ML approach for optimization. The framework is composed of three key processes. Data Distillation collects a significant and representative data set from an objective function. Complexity Reduction ranks the importance and prunes the search space for each variable. Finally, the Enhanced Optimization conducts a search within the reduced space and seeks for the optimal solution.

of extreme values is to enumerate the vertices. These critical points contains rich information about the global optimum.

For a bounded feasible region, the number of vertices is at most combinatorial C_n^d, n being the number of constraints and d the number of variables. By solving any d equations out of n simultaneously, we would obtain a basic solution (if exists). By validating this basic solution with the rest $n-d$ equations, we can check for feasibility of this basic solution. If feasible, this solution is a valid vertex of the search space.

The higher the number of constraints n, the more vertices exist, and building an entire polyhedra becomes a difficult practice. We use Lagrangian relaxation to relax a part of the constraints and result in a loosened polyhedra to draw data points from. Lagrangian relaxation is a common procedure to generate variable bounds during optimization with complex constraints, assuming that a solution to a relaxed problem is an approximate solution to the original problem. Therefore it is safe to claim that a critical point of the relaxed problem is an approximate critical point of the original problem.

We use linear constraints $Ax \leq b$, $A \in \mathbb{R}^{n \times d}$ to illustrate Lagrangian relaxation. During the relaxation, it splits the constraints A such that $A_1 \in R^{n_1 \times d}$, $A_2 \in R^{n_2 \times d}$ and $n_1 + n_2 = n$. Then one introduces part of the constraints into the objective. The problem is converted to a relaxed one:

Min: $f_0(x) + \lambda^T(b_2 - A_2 x)$

s.t.: $A_1 x \leq b_1$

We let $\lambda = (\lambda_1, \ldots, \lambda_{n_2})$ be nonnegative weights so that we are penalized if the relaxed constraints are violated.

In the example of G1, with Lagrangian relaxation we first split the constraint set into two, by randomly selecting 22 out of the 35 inequalities and formulating them as $A_2 x \leq b_2$. Then, the 22 inequalities are removed from the constraint set and the part $(b_2 - A_2 x)$ is plugged into the objective function. The new reduced constraint set then contains exactly 13 inequalities, the same number as that of the variables. Vertex enumeration is performed by trying to solve these remaining 13 equations simultaneously to obtain a basic solution (if exists). This solution is a vertex of the feasible region of the relaxed problem and an approximation of the original problem. This process is repeated 1,000 times, each time with a different random draw of the inequalities. λ is set to be a random vector with values between 0.5 and 1 each time. A higher value of λ suggests a tighter relaxation and a longer time to obtain data.

C. Complexity Reduction with Machine Learning

The centerpiece of our method is reducing the complexity of problem, in two-fold. Firstly, the d variables form an order by feature ranking techniques. Secondly, the searchable region of each variable can be reduced (or formed if that variable is unbounded) by classification schemes.

1) Path Refinement: Feature selection methods in data mining study the variable relations towards the target, either through calculating a metric, or building a classifier. With the exemplar problem G1, we take the data set generated and label the very high values as "H" and low values as "L" (for example, sort the instances based on the function value and take the top and bottom 15%). Then we run six feature selection methods: 1) Information gain, InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute) where H specifies the entropy, 2) Chi-square, $\chi^2 = \sum(O-E)^2/E$, where O is the observed frequency and E is the expected frequency, 3) Symmetrical uncertainty, SU(Class, Attribute) = 2(H(Class) -H(Class | Attribute)) / H(Class) + H(Attribute), where H specifies the entropy, 4) SVM, by using Support Vector Machine as a classifier to evaluate a feature's worth, 5)OneR, by using OneR [5] as a classifier to evaluate a feature's worth, and 6) RELIEF [6], which evaluates a feature by repeatedly sampling an instance and considering the value of the given feature for the nearest instance of the same and different class. The rationale of choosing these feature selection methods is to have a diverse setting that contains both numeric metrics and classifier-based evaluators. An ensemble vote of their outcomes gives a fairly trustworthy result.

2) Region Reduction: In this subroutine, we use the distilled data set to build a rule-based classifier. As in the path refinement activity, the data instances with relatively high values is represented by the symbol "H" and the contradictory class is labeled as "L". This creates a two-class classification problem. We use rule-based classifiers because they are easily traversed and thresholds are clearly attained, as we will show later. Here, we illustrate the example with a random tree. After a tree is constructed, we look for the leaf nodes with "L" since our purpose is to minimize in the G1 example. We traverse from the root to each of the "L" leaf nodes and write down the rules generated along the path. The number of samples covered by the rule and the accuracy of the rule should also be considered. For the G1 example the most supported rule is: IF $x_7 \geq 0.42$ AND $x_{12} \geq 1.13$ AND $x_9 \geq 0.25$ AND $x_{11} \geq 0.43$ THEN "L". Therefore, the searching regions for these variables are modified to $x_7 \in [0.42, 1]$, $x_{12} \in [1.13, 100]$, $x_9 \in [0.25, 1]$, and $x_{11} \in [0.43, 1]$. The searching effort is thus reduced to a more concentrated area on these variables. Note that the satisfaction of the original constraints/boundary conditions must be considered simultaneously. Compared to the original regions, $x_7, x_9 \in [0, 1]$, $x_{11}, x_{12} \in [0, 100]$, relatively 42%, 1.13%, 25%, 43% of the search region has been reduced.

D. Enhanced Optimization with Reduced Complexity

Our proposed searching strategy adopts the spirit of greedy algorithms, whose purpose is to find a locally optimal solution at every step. In our case, we find the value that optimizes the function for one variable at a time, following the ranked list of variables. The searching strategy on each variable is the line search, a well established optimization technique. This strategy is designed to be easy to implement and robust in handling high-dimensional problems.

Multi-start strategies are incorporated so that on each run, the algorithm starts from a randomly generated initial solution of dimension d in the search space. Then the algorithm iterates until a stopping criterion, that is, convergence, is reached. At each iteration, line search is applied from the top variable in the ranked list proceeding downwards.

III. Experiments and Results

In this section, we conduct numerical experiments on a suite of test problems. Firstly we validate the effectiveness of two steps, data distillation and feature selection, by comparing with benchmark methods. Secondly, the entire framework is validated with 7 scalable benchmark problems, whose number of variables can be determined by users. In all sets of experiments, our method is compared with genetic algorithms (GA). The comparison is based on the value of the solution found (if the true optimum is unknown, as in the synthetic cases) or error between solution found and the true optimum (if known, as in the standard problems). Time is also recorded. Experiments are setup on a Linux Red Hat 4.4.7 system with 32 GB memory and Intel Xeon CPU 2.20 GHz.

A. Validation of Data Distillation

The experiments here is to test the effectiveness of our data distillation procedure. To compare with our vertex based method, the quasi-Monte Carlo NTM model proposed in [7] is used to collect a set of uniformly distributed data. Also, a simple randomization of variables on their feasible regions is used. The evaluation is made on the collected data only and does not extend to further optimization. Certain properties of the data are considered: maximum objective value collected, minimum objective value collected, total number of instances collected within a fixed time (100 seconds). Two more problems from [3], G2 and G7, are added to the test suite. They are selected because of their relatively large dimension, as well as their constraint set being fairly complex. At $n = 20$, the best known for G2 is $f(x^*) = 0.803619$. The best known for G7 is $f(x^*) = 24.3062091$. The comparison on all three problems is seen in Table I. As we can see, in almost every scenario our vertex based data collection method is able to produce more polarized data, that is, data with a higher maximum value and lower minimum value. What's more, the number of instances produced within a same period by the vertex enumeration method is much larger, mainly due to the relaxation of constraints in the process.

G2 Problem:

$$\text{Max:} \quad f(x) = \left|\frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}\right|$$

$$\text{s.t.:} \quad g_1(x) = -\prod_{i=1}^{n} x_i + 0.75 \leq 0,$$
$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0,$$
$$0 \leq x_i \leq 10, i = 1, 2, \ldots, n.$$

G7 Problem:

$$\text{Min:} \quad f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_10 - 7)^2 + 45)$$

$$\text{s.t.:} \quad g_1(x) = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \leq 0,$$
$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0,$$
$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0,$$
$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0,$$
$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0,$$
$$g_6(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0,$$
$$g_7(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0,$$
$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_10 \leq 0.$$
$$-10 \leq x_i \leq 10, i = 1, 2, \ldots, 10.$$

B. Validation of Feature Ranking

To demonstrate the effectiveness of feature selection in producing an importance ranking of the variables and hence a refined searching path, we compared with random searching paths, and evaluate the error between the actual optimum and

G1	Max	Min	Total cases
Vertex	2.3128	-9.77	8656
NTM	2.1134	-3.05	1843
Random	0.9887	-3.32	1457
G2 (n=20)			
Vertex	0.7664	-12.112	166
NTM	0.3555	-16.455	28
Random	/	/	0
G7			
Vertex	198.3321	29.332	9656
NTM	201.4456	36..654	63
Random	200.3432	98.4804	17

TABLE I: Validation of data distillation. Comparing our vertex based method to NTM and randomization.

best answer found, as well as the time taken. The refined path is obtained using our method, and the random path is the best one out of 50 runs. The error $e = f_0(x) - f_0(x^*)$ is presented as the difference between the best solution found by our algorithm and the test function's actual global optimum. The comparison can be seen in Table II. We can see our refined path is more effective in finding the minimum of G1, and more efficient in terms of time consumption.

TABLE II: Validation of feature ranking.

G1	Err. Mean	Err. Var.	Time (s)
Refined path	3.65E-02	5.09E-04	24.55
Random path	4.23E+00	2.10E-02	264.76

C. Standard Test Problems with Scalable Dimension

There is a variety of standardized numerical test problems with scalable dimensions. Most of them are only bounded but unconstrained. Examples of such problems include Ackley function, Rosenbrock Function, Sphere Function, and Zakharov Function [8], to name a few. We use a set of 7 problems, function descriptions shown in Table III.

For each function, experiments are conducted in 100, 500 and 1000 dimensions, separately. Each algorithm is run 25 times and the average error is computed, defined as the difference between the best value known and the best value found. Time consumption is also compared. For error evaluation, the stopping criterion is set as the maximum iterations, which is also the number of fitness evaluations. It is set to be $100 \times d$ where d is the variable dimension. For time evaluation, the stopping criterion is either 1) convergence is achieved or 2) the number of iteration exceeds $100 \times d$. Since the objective functions in these problems are not linear, Linear Programming (LP) is not applicable. Pruned Search (PrS) is compared with genetic algorithms (GA) for each function on each tested variable dimension.

Tables IV-VI show the result for mean value error (Err. Mean), error standard deviation (Err. SD.) computed for 25 runs, for both algorithms on each of the 7 functions. At a dimension of 100, our method beat GA in 2 cases, played even in 2 cases, and achieved a fairly comparable result on the other 3. As the dimension enlarges, these numbers evolve to (3, 2, 2) and (4, 1, 2), respectively. A robustness to high dimensions is therefore observed. The time comparison is seen in Fig. 2. Except at a low dimension on $f3$, $f4$ and $f6$ where the time performance is only slightly worse, in all other cases PrS outperforms GA consistently.

TABLE IV: Error comparison for $d = 100$.

#	PrS		GA	
	Err. Mean	Err. SD.	Err. Mean	Err. SD.
1	4.18E-14	5.99E-15	4.54E-10	6.76 E-12
2	8.22E+01	3.34E+00	8.19E+02	9.24E+03
3	0E+00	0E+00	0E+00	0E+00
4	1.33E-08	2.21E-09	8.33E-09	9.53E-10
5	5.39E-03	1.25E-02	4.33E-03	1.21E-02
6	0E+00	0E+00	0E+00	0E+00
7	0E+00	0E+00	1.43E-01	2.33E-04

TABLE V: Error comparison for $d = 500$.

#	PrS		GA	
	Err. Mean	Err. SD.	Err. Mean	Err. SD.
1	3.12E-06	6.32E-08	5.45E-06	8.66E-08
2	1.61E+03	1.56E+02	1.54E+03	3.64E+03
3	0E+00	0E+00	0E+00	0E+00
4	3.43E-08	2.15E-08	6.42E-08	3.64E-08
5	8.59E-04	6.54E-05	7.75E-04	6.43E-06
6	0E+00	0E+00	0E+00	0E+00
7	0E+00	0E+00	8.54E-03	7.68E-04

TABLE VI: Error comparison for $d = 1000$.

#	PrS		GA	
	Err. Mean	Err. SD.	Err. Mean	Err. SD.
1	4.74E-05	7.24E-08	6.67E-05	6.80E-07
2	2.32E+05	3.34E+02	8.19E+05	9.24E+03
3	0E+00	0E+00	0E+00	0E+00
4	9.32E-05	5.16E-05	2.33E-05	4.53E-08
5	7.36E-06	9.56E-07	4.33E-06	1.21E-07
6	2.11E-03	1.23E-03	3.00E-02	3.32E-03
7	5.53E-04	5.61E-05	7.89E-02	9.53E-04

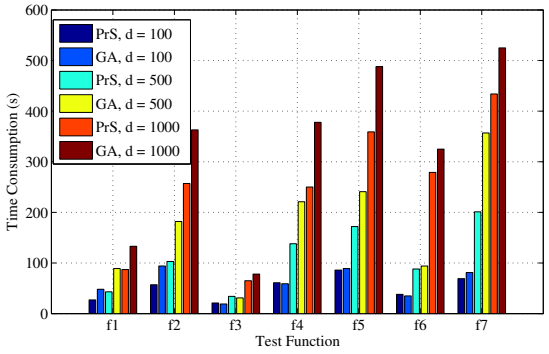

Fig. 2: Comparison of time consumption on standard scalable problems.

IV. Related Works

Some related work using machine learning to assist optimization search are seen. Clustering methods have been used to select promising starting points for optimization algorithms [9] from randomly generated candidate starting points. In [10] Support Vector Machine is employed to learn the relationship between the starting point of an algorithm and the final outcome, so as to bypass the fitness evaluation that

TABLE III: Scalable test functions.

#	Function	$f(x)$	Domain	Optimum
1	Ackley	$20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum x_i^2}} - e^{\frac{1}{d}\sum \cos 2\pi x_i}$	$-32.768 \leq x_i \leq 32.768$	$f(x) = 0, x_i = 0$
2	Rosenbrock	$\sum_{i=1}^{d/2}[100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2]$	$-2.048 \leq x_i \leq 2.048$	$f(x) = 0, x_i = 1$
3	Sphere	$\sum_{i=1}^{d} x_i^2$	$-5.12 \leq x_i \leq 5.12$	$f(x) = 0, x_i = 0$
4	Zakharov	$\sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4$	$-5.0 \leq x_i \leq 10.0$	$f(x) = 0, x_i = 0$
5	Griewank	$1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}})$	$-600.0 \leq x_i \leq 600.0$	$f(x) = 0, x_i = 0$
6	Schwefel	$418.982887272433d - \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i \rvert})$	$-500.0 \leq x_i \leq 500.0$	$f(x) = 0, x_i = 1$
7	Rastrigin	$10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$	$-5.12 \leq x_i \leq 5.12$	$f(x) = 0, x_i = 0$

could be expensive. A new realm of Bayesian global optimization [11] deals with expensive cost functions where evaluating the objective function is costly or even impossible, and the derivatives and convexity properties are unknown. However, those methods only work for low-dimension optimization problems. High dimensionality in optimization problems has been approached by imposing extra hypotheses. In [12] statistical and machine learning ideas are used to change the formulation of the constraints. The setting is that the true constraint parameters lie in a low-dimensional space, but this special structure is obscured by the added noise. In our setup we do not impose hypotheses to the original problem and tend to keep variables exactly as they are.

V. Conclusion

In this paper, we proposed 'Pruned Search', a machine learning approach to enhance the search for optimization problems, considering particularly the ones with a continuous problem space, high dimension of variables and complex constraints. The idea is to construct meta-heuristics to guide the search quest into a pruned, reduced space that is more promising. The meta-heuristics are designed towards a refined search path and trimmed search region. The challenge of machine learning in this problem is that data are not given but need to be generated. The designed data distillation technique based on vertex enumeration and Lagrangian relaxation solves the problem.

We walked through the procedures of our method with an example problem, illustrating each step with intermediate results. Experimental results have shown that our approach can effectively find the global optimal with a significantly reduced computational time compared to Genetic Algorithms on seven high dimension standard benchmark problems.

Interesting future work includes a more thorough investigation of the variable relations. Other than forming a ranked list, for instance, variables may be partial ordered based on multiple, possibly conflicting aspects concerning their influence and impact on the function. They may form subsets, in which superiority might occur. Intercorrelations, dependencies, conditional and causal relations can all be studied. Advances in qualitative decision theory [13]–[16] have shown that conditional dependencies can be used to construct a partial ordering among variables or their subsets efficiently, and it would be interesting to explore the integration of such methods with this work. The resulting knowledge base could provide ground for parallelization in the future, as the reduction of search region for each variable is executed individually. Finally, it would be most interesting to apply the proposed method and its extensions to real-world optimization problems in engineering [17], [18].

Acknowledgement

This work is supported in part by the following grants: AFOSR award FA9550-12-1-0458; NIST award 70NANB14H012; NSF awards CCF-1029166, IIS-1343639, CCF-1409601; DOE award DESC0007456.

References

[1] M. Gen and R. Cheng, *Genetic algorithms and engineering optimization*. John Wiley & Sons, 2000, vol. 7.

[2] R. Liu, A. Agrawal, W.-k. Liao, and A. Choudhary, "Search space preprocessing in solving complex optimization problems," in *2014 IEEE International Conference on Big Data*. IEEE, 2014.

[3] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.

[4] K.-T. Fang and Y. Wang, *Number-theoretic methods in statistics*. CRC Press, 1993, vol. 51.

[5] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine learning*, vol. 11, no. 1, pp. 63–90, 1993.

[6] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*. Morgan Kaufmann Publishers Inc., 1992, pp. 249–256.

[7] K.-T. Fang, Y. Wang, and P. M. Bentler, "Some applications of number-theoretic methods in statistics," *Statistical Science*, pp. 416–428, 1994.

[8] "Test functions for optimization," https://en.wikipedia.org/wiki/Test_functions_for_optimization, [Online; accessed August 2015].

[9] A. R. Kan and G. Timmer, "Stochastic global optimization methods part i: Clustering methods," *Mathematical programming*, vol. 39, no. 1, pp. 27–56, 1987.

[10] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, "Machine learning for global optimization," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, 2012.

[11] E. Brochu, V. M. Cora, and N. 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.

[12] H. Xu, C. Caramanis, and S. Mannor, "Statistical optimization in high dimensions," in *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 1332–1340.

[13] A. Agrawal, "Qualitative decision methods for multi-attribute decision making," *arXiv preprint arXiv:1508.00879*, 2015.

[14] G. R. Santhanam, S. Basu, and V. Honavar, "Representing and reasoning with qualitative preferences for compositional systems," *Journal of Artificial Intelligence Research*, 2011.

[15] G. R. Santhanam, S. Basu, and V. Honavar, "Dominance testing via model checking." in *AAAI*, 2010.

[16] G. R. Santhanam, S. Basu, and V. Honavar, "Efficient dominance testing for unconditional preferences." in *KR*. Citeseer, 2010.

[17] R. Liu, A. Kumar, Z. Chen, A. Agrawal, V. Sundararaghavan, and A. Choudhary, "A predictive machine learning approach for microstructure optimization and materials design," *Nature Scientific Reports*, vol. 5, no. 11551, 2015.

[18] G. R. Santhanam and K. Gopalakrishnan, "Pavement life-cycle sustainability assessment and interpretation using a novel qualitative decision procedure," *Journal of Computing in Civil Engineering*, vol. 27, no. 5, pp. 544–554, 2013.