Search Space Preprocessing in Solving Complex Optimization Problems

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

***Abstract*—**
We look at the complexity placed by big search spaces, dominated by the number of variables and domain of each variable, in search and optimization problems. While a large, even infinite, search domain impairs the effectiveness and efficiency of search, a complex structure of constraints further increases the difficulty in that the search space becomes highly irregular. We propose in this position paper that data mining and dimension reduction techniques have a potential in addressing the pressing issues in both combinatorial optimization and continuous optimization. By preprocessing the original search space, data mining can help boost the speed of search by guiding the search effort to a reduced, more promising area.

***Keywords*-optimization; complexity reduction; data mining; dimension reduction;**

I. Introduction

Big data means more than just the volume of existing, collected data. Complexity can occur from the existence of a large problem space, dominated by the number of features and domains of features, from which decisions, solutions, and answers are to be discovered. A problem space is the stretch of the complete set of data that can possibly happen in this problem. Performing searches in a large problem space is a common practice, whether it is to find a satisfiable solution (e.g. in a constraint satisfaction problem), a best solution (e.g. in an optimization problem), or an answer to a question (e.g. in a decision problem). The larger the problem space, the larger the set of candidate solutions, which can be enumerated (if possible) and presented prior to the search, or generated during runtime along the search. Either way, the challenges incurred by variable/feature dimension, function nonlinearity, structural heterogeneity, etc., which complicate the size and shape and hence obscure the solution space, hinder the search efficiency and effectiveness. In some problems, the solution space is continuous and enumeration is not even an option.

Nearly all learning and inference problems can be formulated as an optimization problem, with a set of variables each having its own domain, finite or infinite, and together forms a problem (or solution) space containing all possible solutions, an objective function that allows us to evaluate the quality of each candidate solution, and optionally, a set of constraints placed upon the variables. In an engineering application, the variables are control parameters that describe the design. The objective function is expressed to stand for a certain system cost, convergence error (to minimize), or profit, stability, utility (to maximize). The constraints of the variables might originate from physical limitations, technical difficulties, economic and feasibility considerations, and so on. An example in chemical engineering sees optimization needs in the design of a chemical plant in the production of polymer gasoline [1]. In automotive engineering, a vehicle simulation model is designed and the objective is to save fuel consumption by controlling battery power, engine speed, motor speed, etc. [2]. Other optimization applications are seen in biological engineering (e.g. optimal enzyme activation in metabolic networks), industrial and production (e.g. composite structure optimization), material informatics (e.g. optimizing a material's property by choosing the right structure), etc. The general goal is to look for a solution (or multiple solutions) within a defined space which produces the desired value of the function. Some special classes of optimization problems can have analytical algorithms developed, such as in linear programming, integer programming, etc., but most of the general optimization problems have to generally rely on the simple and straightforward, search.

We look at two types of search and optimization problems: the combinatorial optimization and the continuous optimization, each can be further complicated with imposed constraints. Both classes of problems can be associated with enormous search space that does not appear to admit tractable solution algorithms exist. Therefore we are forced to rely on searches such as the local search (for combinatorial optimization) and gradient or other heuristic based search (for continuous optimization), and population-based methods such as genetic algorithms and particle swarm optimization.

The efficiency and effectiveness of search can be impaired greatly by the complexity of the search space, which depends on the variable domain, number of variables, and constraint structure. The search strategy adopted also matters, for that among a number of paths towards the real optimum, a good strategy directs to the shortest one.

We propose that concepts from data mining and dimension reduction can be used to preprocess the search space so as to pre-design a premium search path suggesting the least search effort. Data mining and optimization are two inseparable

and collaterally connected areas. Although what we have seen mostly is a data mining problem formulated into an optimization core and ended up using optimization tools to solve for parameters, in this paper we attempt to answer an inverse question: *Can data mining, more specifically, dimension reduction techniques, be used in large scale optimization problems to ease the curse of dimensionality?*

A downright implantation of feature selection into search algorithms can be inappropriate, for that the number of variables in the search problem can hardly be "reduced". The outcome of a search is an assignment, where each variable is associated with a value. Discarding some of the variables, or forming new variables as combinations of old ones, as what common feature selection and extraction methods do, is not going to be applicable, unless proper hypotheses are made, for example, noisy variables exist in the original formulation and it is our purpose to eliminate them. Problems with such hypotheses are not our focus in this paper.

We propose, on the other hand, a preprocessing of the search space of the original problem so as to gain knowledgable insights and *learn* a better searching path than what traditional heuristics offer. The search space preprocessing performs the following major steps:

- Collect, from the problem space, a set of prototypical variable-objective data instances, for data mining purpose.
- Learn, from the collected data, an optimal search order of variables.
- Learn, from the collected data, a reduced domain for each variable, which together forms a reduced search space for the problem.
- Learn, from the collected data, a set of best initial assignments for the search to be restarted with.
- Regard the learned search order and the reduced search space as the new, enhanced circumstance for the search. Perform an informed search with it.

The rest of the paper is organized as follows. We begin by introducing the background and related work of the two areas, optimization and data mining, in Section II and Section III. We also identify key issues in traditional algorithms and present rooms of improvements. Section IV goes into the methodology of the proposed framework, and explains each of our four key processes. Finally in Section V we conclude the paper and specify the key substances of future deployment.

II. Solving Optimization as a Search

Optimization is about finding the "absolute best" decision which corresponds to the minimum (or maximum) of a function, while satisfying certain constraints. Without loss of generality, we focus on maximization problems in this context. Most optimization problems in the world, except for those formulated under special conditions and constraints, have to rely on the act of search to find the solution.

Based on the type of solutions, optimization problems can be roughly divided into discrete optimization, or combinatorial optimization, and continuous optimization. In turn, searches are to perform on either a finite set of solutions, or an infinite space of continuous values. The latter is not necessarily more difficult, since the finite set of the former can be so big that it cannot simply be enumerated. For example, the search of a series of chess moves in order to maximize the winning score.

The two classes of problems share part of the nomenclature as follows. An optimization problem consists of a set of variables $X = \{x_1, \ldots, x_n\}$, a set of domains $D = \{D_1, \ldots, D_n\}$, a set of constraints $C = \{f_1, \ldots, f_m\}$, and an objective function $f_0(X)$. A solution of the problem is an assignment of a value to every variable in X such that every constraint in C is satisfied.

A. Combinatorial Optimization

In a combinatorial optimization problem, domain D_i is a finite set of values that can be assigned to variable x_i. The combined set of assignments, considering the satisfaction of C, forms a solution space Σ of possible solutions σ. The objective function f_0 is able to evaluate the quality of each candidate solution. Our aim is to find the solution that achieves the maximum quality.

$$\sigma^* = \arg\max_{\sigma \in \Sigma} f_0(\sigma). \tag{1}$$

The available means for solving a combinatorial optimization problem depend on the form of the solution space Σ, as well as the type of the objective. For some problems there exists a closed-form expression of the optimum. However, in our study, we focus on the type of problems that do not have efficient algorithms (polynomial time complexity) to find the optimum, either they are NP-hard, or they do not even have any theoretical analysis of their complexity. As a result, they are forced to rely on searches to locate the optimum, where the search space is the same as the solution space Σ of the problem. The search is usually heuristic-based, and usually has no guarantees of actually finding the optimum.

Local search is a class of such algorithms that explores the search space by moving from one candidate solution to another, until a solution deemed optimal is found or a time bound is elapsed. The candidate solution is also called search states. The algorithms keep tract of a current state, and several neighboring states regarded as "similar" to the current one. Neighboring states are generated by making small modifications to the current state. Similar states are viewed as adjacent to each other in the search space. A search path in the search space is formed by taking small steps to move from one current state to a neighboring state which is made the current state at the next step. Variants of local search algorithms, including the hill climbing, beam search, Tabu search [3], etc., mainly differ in one or more of the key procedures:

- How to generate a neighboring state from a current one, more specifically,
  - which variable do we select to modify,
  - how do we modify the selected variable.
- How to escape a suboptimal convergence with, say, restarts.

Each of these key procedures allows room of improvement with data mining tools, as we will explore in the methodology section.

Another popular algorithm for discrete optimization problems is called branch and bound search, which searches the space of partial assignments, beginning with the empty assignment, and assigning variables in X one at a time, in some order. The search space is presented as a tree where internal nodes represent incomplete assignments and leaf nodes stand for complete ones, which may or may not be optimal. The algorithm traverses the tree, in a depth-first manner, keeping the cost of the best solution found so far. For each branch traversed, the upper and lower estimated bounds on the optimal solution is checked, and the branch is discarded if it cannot produce a better solution than the best one found so far by the algorithm. Several key procedures in this algorithm can be further studied by data mining:

- How to select a variable x_i in the step of branching;
- During branching, once selected x_i, in what order each value in D_i is used to extend the current assignment;
- How to estimate a better upper bound and lower bound.

The idea of using data mining to obtain insightful information on these issues is discussed in the methodology section.

B. Continuous Optimization

In a continuous optimization problem, $X \in \mathbb{R}^n$ and the objective and constraint functions $f_i : \mathbb{R}^d \to \mathbb{R}, i = 0, \ldots, m$. Our aim, again, is to maximize f_0 and the problem can be expressed as:

$$\begin{aligned} &\text{maximize} && f_0(X) \\ &\text{subject to} && f_i(X) \leq b_i, i = 1, \ldots, m. \end{aligned}$$

The constants $b_1, \ldots, b_m$ are the limits, or bounds, for the constraints. The domain D_i is also a bound of values, and can now be combined into C. Again, we are looking into problems for which closed-form analytical solutions cannot be found. Moreover, in many practical problems there exist multiple local maxima, which obscure the actual global maxima. Such type of problems is also of our interest of study. The search of maximum in these problems is very analogous to the discrete local search. We begin with an initial X_0, which can be an arbitrary choice, say, a random guess, and move to a better and better solution.

The most common approach is gradient descent, or in the maximization case, gradient ascent. At each update, we take a step in the direction of the steepest ascent, following the slope of the changing f_0, in order to increase the value of f_0. Unlike hill-climbing, in the common gradient ascent all variables are updated at the same time.

Moreover, the performance of gradient ascent depends on the choose of the learning step η. If η is too large, the algorithm overshoots the maximum in each iteration, while too small a η renders the convergence too slow. Thus, rooms of improvement in continuous optimization algorithms are:

- How to choose an initial X_0 intelligently;
- How to use an ordered variable to perform the update;
- How to take into consideration the constraints during the update;
- How to adaptively choose the step size η at each step.

We will discuss the use of data mining techniques to attend to these issues.

III. Data Mining and Dimension Reduction

The literature is rich about using optimization to solve data mining problems, but not so much for the inverse problem: incorporating data mining into optimization searches. Even for the limited related work, the setup is rather different. Clustering methods have been used to select promising starting points for optimization algorithms [4], [5] from randomly generated candidate starting points. In [6] Support Vector Machine (SVM) 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 could be expensive. A new realm of Bayesian global optimization [7] 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.

Dimension reduction is an important task in data mining that tackles problem with large dimensions. High dimensionality in optimization has been approached recently. In [8] statistical and machine learning ideas are used to change the formulation of the constraints with techniques like Principal Component Analysis (PCA). The setting is that the true constraint parameters lie in a low-dimensional space, but this special structure is obscured by the added noise. However in our setup we do not impose hypotheses to the original problem and tend to keep variables exactly as they are.

IV. Search Space Preprocessing

We consider using dimension reduction techniques to preprocess the vast search space, so as to enhance the search process in optimization, particularly in the high dimensional and complex regime. We attempt to have the search force focused in a concentrated, more promising path and prune the irrelevant effort.

The search space of an optimization problem, whether discrete or continuous, is dominated by three factors: the number of variables n, the combined size of domains D of variables, and the structure formed by the constraints. Each

constraint function can be viewed as a hyperplane in the problem space, and together with the objective function they form a polyhedron that is equivalent to the search space. The more complex the structure of constraints, the more irregular in shape the polyhedron and hence the more difficult for a search path to form.

Our proposed search space preprocessing contains four major components. The functionalities of each component are introduced below, and further explained in following subsections, with regards to the two classes of optimization problems.

1) *Dataset construction*. For data mining techniques to work, we first need to have access to a set of data that contain valid variable assignments and the corresponding objective function value (or goodness of it). In such a dataset, each instance is an assignment along with the evaluation score of the assignment. Each column accounts for a variable x_i, and an additional column acts as the class label in supervised learning.
2) *Search order reduction*. With feature selection methods in dimension reduction, an ordered list of variables can be formed based on their influence and impact towards the function value. To search with such an informed order one has more advantage than to search with random, or no order.
3) *Search domain reduction.* Each variable has its own domain specified, which can be reduced by means of data mining. A smaller search domain theoretically speeds up the search.
4) *Initial point selection.* The initial point at restarting, a general procedure in search to avoid local optima, is often determined by a random guess, which has no guarantee of ending up with yet another local optima. Data mining can help determining a better initial point, by learning from a collection of multiple initial points and where they end.

After search space preprocessing completes with these four steps, search becomes a much promising endeavor, for that the space is reduced and a pre-planned searching path is deployed. The following subsections explains in more detail, the design of each step and the application in the two classes of optimization in concern.

A. Dataset Construction

This step can be regarded as a preparation for the study of the search space. Simply we need data to perform data mining. For discrete problems, such a data can be a subset of the solution set Σ, only each solution σ will be associated with a score, given by an evaluation of the objective function $f_0(\sigma)$. For continuous problems, obtaining such a data requires drawing points from the polyhedra of feasible space.

Although the solution set and the problem space are there, ready for us to draw instances from, the construction of a useful data set is not as easy as one would speculate. On the one hand, we want the data to be distinguishably significant in that it contains instances with the most wanted (highest or lowest depending on whether it is maximization or minimization) function values. Randomized instances are not as representative. A proposed solution is to transform the problem into the regime of computational geometry, where a polyhedra represents the feasible space, and the vertices of the polyhedra are extreme values of the function. Therefore, to obtain a set of extreme values is to extract the vertices. This procedure is called vertex enumeration [9] in computational geometry.

On the other hand, the acquisition of assignments can be obstructed by the complexity in constraints. When the number of constraints m is high, the number of vertices is high. In this case, building a representation of the entire feasible polyhedra is not an easy exercise. We propose to use Lagrangian relaxation to relax a part of the constraints and construct a loosened polyhedra to place data points. Lagrangian relaxation is a common procedure to generate variable bounds during optimization with complex constraints. It is generally accepted that a solution to the 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, and should provide as useful information.

B. Search Order and Domain Reduction

The essential idea of preprocessing the search space is to reduce it. Therefore, these two-fold reduction is the centerpiece of the proposed work. First of all, the n variables can form a ranked list as a result of feature selection. Second, the searchable domain of each variable can be reduced (or formed if that variable is unbounded) by classification schemes.

Feature selection methods in data mining, such as Information Gain, Chi-square, and SVM evaluator, study the variable relations towards the target, through either calculating a metric or building a classifier. Some outputs a rank of the variables which can be used as an ordered searching path.

The search domain of each variable can be reduced by building a rule-based classifier with data labeled from two classes, with relatively high values represented by the symbol "H" and the contradictory class labeled as "L". We propose to use rule-based classifiers because they are easily traversed and thresholds are clearly attained. After a classifier is constructed, we look for the rules with "H" if our purpose is to maximize. Each rule should contain "IF" clauses with variables and value thresholds. The thresholds of variables in the critical rules can be used as new bounds for the corresponding variable.

C. Initial point selection

Restarting is a common procedure in a lot of search algorithms, especially for non-convex problems with mul-

tiple local maxima. Aiming to escape a plateau caused by local maxima, one often restarts with another initial point to conduct a new search. The initial point is often determined rather carelessly, by a simple randomization. We argue that a learned, verified, deliberately selected initial point to start the search with can be of great value to the search result and efficiency.

Such a learning can be performed on a collected data set of multiple starting points of assignments, each with (the goodness of) where and how it ends. Classification models can be trained to differentiate good starting points from bad ones, therefore during the search, the selection of starting points can become well informed, and redundant and futile search paths can be avoided.

V. Conclusion

Search and optimization in big data realm is becoming more and more challenging due to the increasing size and complexity of the problem space, which indicates an enormous number of candidates of solution, contained in a highly obscured irregular space. The high dimensions in variables and large complexity in constraints can render traditional search methods time consuming and ineffective.

Optimization has been a principal act in machine learning and data mining ever since the very beginning of development of regression analysis back in the 1800s. For example, linear regression is about finding the right coefficient assignment to optimize a least squares error. The act of search has been a general procedure in solving optimization problems where an analytical solution can hardly be found. As an interesting turn of events, we consider that data mining can also serve to assist the optimization search by preprocessing the search space, with variable relations analysis, search space reduction, searching path refinement, and initial condition validation.

This position paper pursues precisely this avenue, with a structure that incorporates ideas from computational geometry, machine learning, data mining and dimension reduction. The proposed method, search space preprocessing, is to be deployed prior to the search, to construct an enhanced condition by reordering the search path, reducing the search domain, and knowledgeably select initial points to restart from.

Our approach attempts to reduce the search space and direct the searching effort to a concentrated, more promising area. Unlike a usual data mining problem where data is always given, what we have in an search and optimization problem is only a constrained function. To be able to perform data mining techniques to gain informative insights, we design a data construction technique based on vertex enumeration and Lagrangian relaxation, so that significant and representative data instances are attained. Based on the constructed data, we design a strategy for refining the search path and reducing the search region using feature selection and classification techniques. For searches that require multiple restarts to escape from local optima, we provide means to differentiate good starting points from bad ones, so that one can choose informedly each round of restart.

To successfully deploy the designed approach in future work, one needs to verify the effectiveness of each of the four steps with empirical study. For example, is a polarized data set, containing only instances of "extremely high" and "extremely low" objective values a better learning base than, say, a random set? Is the variable order generated by feature selection meaningful? Which feature selection should we use under different circumstances? In using rule-based classifiers to extract a reduced domain for each variable, how to deal with contradictory bounds from different rules? And finally, how much improvement in search speed and accuracy considering the overhead of preprocessing is worth evaluation.

Acknowledgment

This work is supported in part by the following grants: NSF awards CCF-1029166, ACI-1144061, IIS-1343639, and CCF-1409601; DOE award DESC0007456; AFOSR award FA9550-12-1-0458; NIST award 70NANB14H012.

References

[1] P. Friedman and K. Pinder, "Optimization of a simulation model of a chemical plant," *Industrial & Engineering Chemistry Process Design and Development*, vol. 11, no. 4, pp. 512–520, 1972.

[2] J. Park, Z. Chen, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey, "Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 9, pp. 4741–4756, 2009.

[3] F. Glover and M. Laguna, *Tabu search*. Springer, 1999.

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

[5] ——, "Stochastic global optimization methods part ii: Multi level methods," *Mathematical Programming*, vol. 39, no. 1, pp. 57–78, 1987.

[6] 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.

[7] 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.

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

[9] H. Edelsbrunner, *Algorithms in combinatorial geometry*. Springer, 1987, vol. 10.